NSWC TR 80-18

PROGRAM DESIGN LANGUAGE ARCHITECTURE SPECIFICATION FOR THE AN/UYK-7 CENTRAL PROCESSOR

by
ALAN HYNSON
JAY MEYERS
CHARLES NAPLES
Strategic Systems Department

FEBRUARY 1980

NAVAL SURFACE WEAPONS CENTER

Dahlgren, Virginia 22448 Silver Spring, Maryland 20910

NAVAL SURFACE WEAPONS CENTER
Dahlgren, Virginia 22448

Paul L. Anderson, Capt., USN
Commander

UNCLASSIFIED

SECURITY CLASSIFICATION OF THIS PAGE (When Data Entered)

REPORT DOCUMENTATION PAGE

READ INSTRUCTIONS BEFORE COMPLETING FORM

1. REPORT NUMBER: NSWC/TR-80-78	2. GOVT ACCESSION NO.: AD-A084 022
3. RECIPIENT'S CATALOG NUMBER:	
4. TITLE (and Subtitle): PROGRAM DESIGN LANGUAGE ARCHITECTURE SPECIFICATION OF THE AN/UYK-7 CONTROL PROCESSOR for Central	5. TYPE OF REPORT & PERIOD COVERED: Final Rept's
	6. PERFORMING ORG. REPORT NUMBER:
7. AUTHOR(s): Alan J. Hynson, Jay C. Meyers, Charles J. Naples	8. CONTRACT OR GRANT NUMBER(s): 12. 3.00
9. PERFORMING ORGANIZATION NAME AND ADDRESS: Naval Surface Weapons Center (K74), Dahlgren, VA 22448	10. PROGRAM ELEMENT, PROJECT, TASK AREA & WORK UNIT NUMBERS: NIF
11. CONTROLLING OFFICE NAME AND ADDRESS: Naval Surface Weapons Center (K74), Dahlgren, VA 22448	12. REPORT DATE: Feb 1980
	13. NUMBER OF PAGES: 203
14. MONITORING AGENCY NAME & ADDRESS (if different from Controlling Office)	15. SECURITY CLASS. (of this report): Unclassified
	15a. DECLASSIFICATION/DOWNGRADING SCHEDULE

16. DISTRIBUTION STATEMENT (of this Report)

Approved for public release; distribution unlimited.

17. DISTRIBUTION STATEMENT (of the abstract entered in Block 20, if different from Report)

18. SUPPLEMENTARY NOTES

19. KEY WORDS (Continue on reverse side if necessary and identify by block number)

Program Design Language (PDL); emulation, computer architecture, microprogramming, AN/UYK-7, central processor (CP)

20. ABSTRACT (Continue on reverse side if necessary and identify by block number)

The Univac AN/UYK-7 computer is the Navy's standard mainframe computer for medium and large scale applications. This report presents a Program Design Language (PDL) description of the AN/UYK-7 Central Processor (CP) architecture and is intended for use as a reference document.

This PDL was used as a design specification for a microprogrammed emulation of the AN/UYK-7 Central Processor. This emulation has been (continued)

DD FORM 1 JAN 73 1473 EDITION OF 1 NOV 65 IS OBSOLETE
S/N 0102-LF-014-6601

UNCLASSIFIED
SECURITY CLASSIFICATION OF THIS PAGE (When Data Entered)

411567

validated through the successful execution of the ~~Univac-supplied~~ AN/UYK-7 Central Processor (CP) diagnostics and by an independent testing effort.

FOREWORD

This document contains a comprehensive description of the central processor architecture of the Navy Standard AN/UYK-7 computer. The architecture was described using a Program Design Language (PDL) supplied by Caine, Farber, and Gordon, Inc. As far as can be determined this document contains the most accurate description of the central processor of the AN/UYK-7 computer published to date.

The PDL was used in the development of a microprogrammed AN/UYK-7 central processor emulation on a Nanodata QM-1 computer. This emulation has been validated through the successful execution of the UNIVAC-supplied AN/UYK-7 CP diagnostics and by an independent testing effort.

The authors gratefully acknowledge the efforts of Henry Walker (SPERRY UNIVAC) and Marc Hubbard (Gun Fire Control Systems Branch, Combat Systems Department) for providing assistance in clarifying ambiguous UNIVAC documentation and verification of architectural questions on actual AN/UYK-7 hardware and Helen Fletcher for assistance in the preparation of this document.

This report was prepared in the Programming Systems Branch of the Computer Programming Division and reviewed by Hermon W. Thombs, Head, Programming Systems Branch.

Released by:

R. T. RYLAND, JR., Head
Strategic Systems Department

TABLE OF CONTENTS

TABLE OF CONTENTS

TABLE OF CONTENTS

AN/UYK-7 CP SEQUENCES

CP MAIN LOOP ..MAIN LOOP FOR AN/UYK-7 CENTRAL PROCESSOR

```
REF
PAGE ********************************************************************************************************************
     *                                                                                                                  *
14   *  1    DO FOREVER ..LOOP THROUGH INSTRUCTION, INDIRECT ADDRESS, OPERAND & INTERRUPT SEQUENCES.                     *
16   *  2        IF _STEP SET, THEN                                                                                      *
17   *  3            SUSPEND LOOPING UNTIL RESTARTED                                                                     *
18   *  4        ENDIF                                                                                                   *
19   *  5        IF ASR(13,1) "CLASS II LOCKOUT SET" .OR.                                                                *
20   *  6            ASR(11,1) "INTERRUPT BASE REGISTERS SELECTED" .OR.                                                  *
21   *  7            ASR(9,1) "MEMORY LOCKOUT INHIBIT SET", THEN                                                         *
22   *  8            DISABLE SPR_CHECKS                                                                                  *
23   *  9        ELSE                                                                                                    *
24   * 10            ENABLE SPR_CHECKS                                                                                   *
25   * 11        ENDIF                                                                                                   *
26   * 12        IF NOT REPEAT_IN_PROGRESS, THEN                                                                         *
27   * 13            CLEAR MEMORY_STORE_INDICATOR ..THIS INDICATOR IS SET BY INSTRUCTIONS                                *
28   * 14            ..THAT STORE INTO MEMORY. IT IS USED IN THE REPEAT TERMINATION LOGIC.                               *
29   4 * 15            CALL I_SEQUENCE ..FETCH INSTRUCTION, PERFORM BREAKPOINT & SPR CHECKS.                           *
31   * 16        ENDIF                                                                                                   *
32   * 17        IF INTERRUPT_SCAN_INHIBIT CLEAR, THEN                                                                   *
33   5 * 18            CALL INTERRUPT_SCAN ..CHECK FOR ASYNCHRONOUS INTERRUPTS.                                        *
34   * 19        ENDIF                                                                                                   *
35   * 20        CALL _DECODE ..EXECUTE INSTRUCTION INDICATED BY U(FUNCTION DESIGNATORS).                                *
37   * 21        IF REPEAT_IN_PROGRESS, THEN                                                                             *
38   9 * 22            CALL REPEAT_SEQUENCE ..PERFORM REPEAT TERMINATION LOGIC.                                        *
39   * 23        ELSE                                                                                                    *
40   * 24            IF ASR(15,1) "LOWER HALF-WORD INSTRUCTION", THEN                                                    *
41   * 25                SET INTERRUPT_SCAN_INHIBIT ..DON'T ALLOW ASYNCHRONOUS INTERRUPTS                                *
42   * 26                ..BETWEEN HALF-WORD INSTRUCTIONS.                                                               *
43   * 27            ELSE                                                                                                *
44   * 28                CLEAR INTERRUPT_SCAN_INHIBIT                                                                    *
45   * 29            ENDIF                                                                                               *
46   * 30        ENDIF                                                                                                   *
47   * 31        IF CP MONITOR CLOCK POSITIVE (I.E. >=0), THEN                                                           *
48   * 32            DECREMENT CP MONITOR CLOCK EACH 1/1024 SECOND                                                       *
49   * 33            IF CP MONITOR CLOCK NEGATIVE, THEN                                                                  *
50   * 34                GENERATE CP MONITOR CLOCK INTERRUPT TO BE DETECTED BY INTERRUPT_SCAN                            *
52   * 35            ENDIF                                                                                               *
53   * 36        ENDIF                                                                                                   *
54   * 37    ENDDO                                                                                                       *
     *                                                                                                                  *
     ********************************************************************************************************************
```

```
I_SEQUENCE

REF
PAGE ********************************************************************************************************************
     *                                                                                                                  *
56   *  1    .. MOVE THE NEXT INSTRUCTION FROM MEMORY INTO THE U REGISTER.                                              *
57   *  2    ..PERFORM APPROPRIATE BREAKPOINT AND SOFTWARE PROTECTION REGISTER (SPR) CHECKS.                            *
59   *  3    32_REG(3) := P(0) - ASR(15,1) ..THE PROPER DISPLACEMENT TO THE NEXT INSTRUCTION.                          *
61   *  4    .. REFETCH U ON A LOWER HALF-WORD SINCE A SYNCHRONOUS INTERRUPT MAY                                        *
62   *  5    ..HAVE BEEN PROCESSED SINCE EXECUTION OF THE UPPER HALF-WORD INSTRUCTION.                                  *
64   *  6    32_REG(1) := 32_REG(3) ..GET TEMPORARY COPY OF DISPLACEMENT.                                              *
65  32*  7    CALL ADD_S (P(S), 32_REG(1)) ..CONDITIONALLY ADD IN S REGISTER.                                           *
66   *  8    IF ASR(15,1)=0 "NOT EXECUTING LOWER HALF OF INSTRUCTION", THEN                                            *
67  17*  9        CALL BPR_CHECK (32_REG(1), INSTRUCTION ADDRESS) ..CHECK FOR BREAKPOINT.                               *
68   * 10    ENDIF                                                                                                     *
69  20* 11    CALL SPR_CHECK (P(S), INSTRUCTION EXECUTE, "P-"0, 32_REG(3))                                              *
70  19* 12    CALL MEMORY_READ (32_REG(1), U, P(S), INSTRUCTION, "P-"0)                                                 *
71      * 13    IF ASR(15,1) SET "EXECUTING LOWER HALF-WORD INSTRUCTION", THEN                                        *
72      * 14        UU := UL ..SHIFT LOWER HALF-WORD INSTRUCTION                                                      *
73      * 15    ENDIF                                                                                                 *
74      * 16    CLEAR EXECUTE_REMOTE_IN_PROGRESS                                                                      *
75      * 17    CLEAR CHARACTER_ADDRESSING_OVERIDE                                                                    *
76      * 18    CLEAR SPR_PRIVILEGED_INSTRUCTION ..ASSUME NO SPECIAL PRIVILEGED INSTRUCTION CHECKS                    *
78      * 19    P(C) := 32_REG(3)+1 ..INCREMENT P TO NEXT INSTRUCTION ADDRESS.                                        *
79      * 20    DEFINE INSTRUCTION_FORMAT_INDIACTOR ..I.E. I, II, III, OR IV.                                         *
80      * 21    IF REPEAT_PENDING, THEN                                                                               *
81      * 22        IF INSTRUCTION REPEATABLE (AS PER REPERTOIRE CARD), THEN                                          *
82      * 23            SET REPEAT_IN_PROGRESS                                                                        *
83      * 24            CLEAR REPEAT_PENDING                                                                          *
84      * 25            SET SPR_PRIVILEGED_INSTRUCTION ..ALL REPEATED INSTRUCTIONS PRIVILEGED WHEN                    *
86      * 26            ..SPR(16,1) SET "INTERRUPT BES FOR INDIRECT ADDRESSING" AND INDIRECT ADDRESSING.              *
88      * 27            IF 60>U(F)>53 "CMR REFERENCE INSTRUCTION", THEN                                               *
89      * 28                SET INTERRUPT_SCAN_INHIBIT ..DON'T ALLOW DETECTION OF ASYNCHRONOUS INTERRUPTS             *
91      * 29                ..WHEN REPEATING A CMR REFERENCE INSTRUCTION.                                             *
92      * 30            ENDIF                                                                                         *
93      * 31        ELSE                                                                                              *
94      * 32            CLEAR REPEAT_PENDING                                                                          *
95      * 33            ABORT THE INSTRUCTION                                                                         *
96      * 34        ENDIF                                                                                             *
97      * 35    ENDIF                                                                                                 *
98      * 36    RETURN                                                                                                *
     *                                                                                                                  *
     ********************************************************************************************************************
```

```
     INTERRUPT_SCAN

     REF
     PAGE ********************************************************************************************************************
          *                                                                                                                    *
100       *  1    .. THE ASYNCHRONOUS INTERRUPT SCAN MECHANISM WILL NOT BE PERMITTED                                          *
101       *  2    ..UNDER THE FOLLOWING SITUATIONS:                                                                           *
102       *  3    .. (1) DURING REPEATS OF FUNCTION CODES 54-57.                                                              *
103       *  4    .. (2) BETWEEN EXECUTION OF UPPER & LOWER HALFWORD INSTRUCTIONS.                                            *
104       *  5    SET _CODE INVALID ..INITIALIZE INTERRUPT STATUS CODE.                                                       *
105       *  6    IF CLASS I INTERRUPT PENDING .AND. ASP(14,1)=0 "CLASS I NOT LOCKED OUT", THEN                               *
107       *  7        _CLASS := I ..DEFINE INTERRUPT CLASS                                                                    *
108    12 *  8        CALL GET_ISCI_CLASS,_CODE) ..GET INTERRUPT STATUS CODE FROM IOC                                         *
109       *  9    ELSE                                                                                                        *
110       * 10        IF CP_MONITOR_CLOCK_INTERRUPT PENDING .AND. ASP(13,1)=0 "CLASS II NOT LOCKED OUT", THEN               *
112       * 11            _CODE := B'1111' ..DEFINE INTERRUPT STATUS CODE                                                     *
113       * 12            _CLASS := II ..DEFINE INTERRUPT CLASS                                                               *
114       * 13            CLEAR CP_MONITOR_CLOCK_INTERRUPT INDICATOR                                                          *
115       * 14        ELSE                                                                                                    *
116       * 15            IF CLASS III INTERRUPT PENDING .AND. ASP(12,1) =0 "CLASS III NOT LOCKED OUT", THEN                  *
118       * 16                _CLASS := III ..DEFINE INTERRUPT CLASS                                                          *
119    12 * 17                CALL GET_ISCI_CLASS,_CODE) ..GET INTERRUPT STATUS CODE FROM IOC.                                *
120       * 18            ELSE                                                                                                *
121       * 19                IF INTERPROCESSOR_INTERRUPT PENDING .AND. ASP(13,1)=0 "CLASS II NOT LOCKED OUT", THEN           *
123       * 20                    _CODE := B'0000' ..DEFINE INTERRUPT STATUS CODE.                                            *
124       * 21                    _CLASS := II ..DEFINE INTERRUPT CLASS.                                                      *
125       * 22                    CLEAR INTERPROCESSOR_INTERRUPT INDICATOR                                                    *
126       * 23                ENDIF                                                                                           *
127       * 24            ENDIF                                                                                               *
128       * 25        ENDIF                                                                                                   *
129       * 26    ENDIF                                                                                                       *
130       * 27    IF _CODE VALID, THEN ..THE IOC MAY RETURN AN INVALID CODE.                                                  *
131       * 28        IF REPEAT_IN_PROGRESS, THEN                                                                             *
132       * 29            P := P-2 ..I.E. ADDRESS OF REPEAT INSTRUCTION.                                                      *
133       * 30        ELSE                                                                                                    *
134       * 31            P := P-1 ..SET P TO REEXECUTE CURRENT INSTRUCTION.                                                  *
135       * 32        ENDIF                                                                                                   *
136       * 33        CLEAR REPEAT_IN_PROGRESS AND REPEAT_PENDING INDICATORS                                                  *
137     8 * 34        CALL INTERRUPT_SEQUENCE (_CLASS, _CODE)                                                                 *
138       * 35    ENDIF                                                                                                       *
139       * 36    RETURN                                                                                                      *
          *                                                                                                                    *
          ********************************************************************************************************************
```

_DECODE

```
141     +  1    .. DECODE INSTRUCTION IN U BASED  I U(F). SOME INSTRUCTIONS REQUIRE
142     +  2    ..SUBFUNCTION FIELDS (SEE DECODE_.HASE_TWO).
143     +  3    IF U(S)=7 .AND. ASR(8,1)=1 .ANC.(.NOT. LBMP INSTRUCTION) .AND. TASK
144     +  4         MODE, THEN
145     +  5      CALL GENERATE_SYNCHRONOUS_INTERRUPT
146     +  6           ("P-"1, PRIVILEGED INSTRUCTION, DUMPY_CODE) ..ABORT INSTRUCTION.
147     +  7    ENDIF
148     +  8
149     +  9    DO CASE U(F) CORRESPONDING TO THE FOLLOWING TABLE
150     + 10
151     + 11     ILLEGAL,     F01,     F02,     F03, ILLEGAL,     F05,     F06,     F07,
152     + 12
153     + 13         _LA,    _LXB,   _LDIF,    _ANA,     _AA,   _LSUM,    _LNA,     _LM,
154     + 14
155     + 15         _LB,     _AB,    _ANM,     _SB,     _SA,    _SXB,    _SNA,     _SM,
156     + 16
157     + 17     ILLEGAL, ILLEGAL,  _BZ_BS,  _BZ_BS,     _RA,     _RI,    _RAN,     _RC,
158     + 18
159     + 19          _M,      _D,      _BC,    _CXI,      _C,     _CL,     _CM,     _CG,
160     + 20
161     + 21         F5C,    _F51,     F52,     F53,    _LCT,    _LCI,    _SCT,    _SCI,
162     + 22
163     + 23      HSC_6C,  HLC_61,    _HLC,   _HDLC,   _HRZ,   _HDRZ,    _HPS,   _HDPS,
164     + 24
165     + 25         F70,     F71, ILLEGAL, ILLEGAL,     F74, ILLEGAL, ILLEGAL,     F77
166     + 26
167     + 27    ENDDO
168     + 28
169     + 29     ILLEGAL:
170     + 30       CALL GENERATE_SYNCHRONOUS_INTERRUPT
171     + 31            ("P-"1, CP ILLEGAL INSTRUCTION, DUMMY_CODE) ..ABORT INSTRUCTION.
172     + 32
```

DECODE_PHASE_TWO

```
174        +  1     NOTE:  ____ -> ILLEGAL INSTRUCTION.
175        +  2
176        +  3     F01    :  DO CASE U(F2) CORRESPONDING TO THE FOLLOWING TABLE
177        +  4                 _OR,  _SC,  _RS, _XOR, _ALP, _LLP, _NLP, _LLPN
178        +  5
179        +  6     F02    :  DO CASE U(F2) CORRESPONDING TO THE FOLLOWING TABLE
180        +  7               _CNT, ____,  _XP, _XPL, _SLP,_SSUM,_SDIF,   _DS
181        +  8
182        +  9     F03    :  DO CASE U(F2) CORRESPONDING TO THE FOLLOWING TABLE
183        + 10               _ROR, _RSC, _RRS,_RXOR,_PALP, _RLP,_RNLP,  _TSF
184        + 11
185        + 12     F05,   :  DO CASE U(F2) CORRESPONDING TO THE FOLLOWING TABLE
186        + 13                 _DL,  _DA, _DAN,  _DC,_LEMP, ____,  ____,  ____
187        + 14
188        + 15     F06,   :  DO CASE U(F2) CORRESPONDING TO THE FOLLOWING TABLE
189        + 16                 _FA, _FAN,  _FM,  _FD, _FAR,_FANR, _FMR,  _FDP
190        + 17
191        + 18     F07    :  DO CASE U(F2) CORRESPONDING TO THE FOLLOWING TABLE
192        + 19               F070, _AEI, _PEI, _LIP,  _IO,  _IR,  _RP,  ____
193        + 20
194        + 21     F070   :  DO CASE U(A+2,1) CORRESPONDING TO THE FOLLOWING TABLE
195        + 22                 _RS,  _IPI
196        + 23
197        + 24     F50    :  DO CASE U(F3) CORRESPONDING TO THE FOLLOWING TABLE
198        + 25               _JEP, _JOP, _DJZ,_DJNZ
199        + 26
200        + 27     F52    :  DO CASE U(F3) CORRESPONDING TO THE FOLLOWING TABLE
201        + 28               _LBJ,_JBNZ,  _JS,  _JL
202        + 29
203        + 30     F53    :  DO CASE U(F3) CORRESPONDING TO THE FOLLOWING TABLE
204        + 31              _J530,_J531,_J532,_J533
205        + 32
206        + 33     F70    :  DO CASE U(F4) CORRESPONDING TO THE FOLLOWING TABLE
207        + 34               _HSF,_HDSF, _HCP,_HDCP, ____,  ____,  ____,  ____
208        + 35
209        + 36     F71    :  DO CASE U(F4) CORRESPONDING TO THE FOLLOWING TABLE
210        + 37               _HOR,  _HA, _HAN,_HXCH, ____,_HAND,  ____,  ____
211        + 38
212        + 39     F74    :  DO CASE U(F4) CORRESPONDING TO THE FOLLOWING TABLE
213        + 40                 _HM,  _HD, _HRT, _HLB,  _HC, _HCL, _HCM,  _HCB
214        + 41
215        + 42     F77    :  DO CASE U(F4) CORRESPONDING TO THE FOLLOWING TABLE
216        + 43              _HSIM,_HSTC,  ____,  ____, _HPI, _HAI, H776,  ____
```

```
     INTERRUPT_SEQUENCE (_CLASS, _ISC)

REF
PAGE ********************************************************************************************************************
     *                                                                                                                  *
218  *  1    .. P MODIFICATION PERFORMED PRIOR TO ENTRY.                                                               *
219  *  2    ..THERE IS NO PROVISION FOR POWER TOLERANCE INTERRUPTS (I.E.                                               *
220  *  3    ..NEVER OCCUR, THUS NEVER HANDLED) IN THIS EMULATION.                                                     *
221  *  4    .. (1) _CLASS - THE CLASS (LEVEL) OF INTERRUPT BEING PROCESSED.                                           *
222  *  5    .. (2) _ISC - INTERRUPT STATUS CODE ASSOCIATED WITH THIS INTERRUPT.                                       *
223  *  6    ASR -> CPR (O'135' + 4*_CLASS) ..SAVE ACTIVE STATUS REGISTER.                                             *
224  *  7    _ISC -> CMR (O'136' + 4*_CLASS) ..SAVE INTERRUPT STATUS CODE.                                             *
225  *  8    P -> CMR (O'137' + 4*_CLASS) ..SAVE PROGRAM COUNTER.                                                      *
226  *  9    DO CASE _CLASS OF                                                                                         *
227  * 10        \I\ ..MUST BE NON-POWER TOLERANCE (SEE COMMENT ABOVE).                                                *
228  * 11        ASR(19,20):=D'2077600' ..LEAVE CPID. SET STATE I, CLASS I,II,III LOCKOUTS.                            *
230  * 12        ..SET INTERRUPT S,B SELECTS, MEMORY LOCKOUT INHIBIT, LBMP ENABLE AND BOOTSTRAP.                       *
232  * 13        P(S) := 7; P(0) := 0 ..FORCE NORO SWITCHER ACTIVATION.                                                *
233  * 14        \II\                                                                                                  *
234  * 15        ASR(19,20) := ASR(19,20).A.O'40200' ..LEAVE CPID, CLASS I,LOCKOUTS,                                   *
235  * 16        ..AND BOOTSTRAP BITS UNCHANGED.                                                                       *
236  * 17        ASR(19,20) := ASR(19,20).V.O'137400' ..SET STATE II,CLASS II,III LOCKOUTS.                            *
238  * 18        ..SET INTERRUPT S,B SELECTS, MEMORY LOCKOUT INHIBIT & LBMP ENABLE.                                    *
239  * 19        IF AUTO_RECOVERY SELECTED .AND. _ISC=B'001C' "ILLEGAL INSTRUCTION", THEN                              *
241  * 20            ASR(7,1) := 1 ..SET BOOTSTRAP MODE.                                                               *
242  * 21            P(S) := 7; P(0) := 1 + BOOTSTRAP_SWITCH SETTING ..(I.E. 1,2,OR 3)                                 *
243  * 22        ELSE                                                                                                  *
244  * 23            P:= CMR(O'144') ..CLASS II ICW.                                                                   *
245  * 24        ENDIF                                                                                                 *
246  * 25        \III\                                                                                                 *
247  * 26        ASR(19,20) := ASR(19,20).A.O'60200' ..LEAVE CPID, CLASS I & II LOCKOUT,                               *
249  * 27        ..AND BOOTSTRAP BITS UNCHANGED.                                                                       *
250  * 28        ASR(19,20) := ASR(19,20).V.O'417400' ..SET STATE III, CLASS III LOCKOUT.                              *
252  * 29        ..SET INTERRUPT S,B SELECTS, MEMORY LOCKOUT INHIBIT & LBMP ENABLE.                                    *
253  * 30        P := CPR(O'150') ..CLASS III ICW.                                                                     *
254  * 31        \IV\                                                                                                  *
255  * 32        ASR(19,20) := ASR(19,20).A.O'70200' ..LEAVE CPID, BOOTSTRAP BIT AND LOCKOUTS UNCHANGED.               *
257  * 33        ASR(19,20) := ASR(19,20).V.O'207400' ..SET STATE IV.                                                  *
258  * 34        ..SET INTERRUPT S,B SELECTS, MEMORY LOCKOUT INHIBIT & LBMP ENABLE.                                    *
259  * 35        P := CPR(O'154') ..CLASS IV ICW.                                                                      *
260  * 36    ENDDO ..END CASE.                                                                                         *
261  * 37    RETURN ..TO END OF MAIN LOOP (ABORTING INSTRUCTION).                                                      *
     *                                                                                                                  *
     ********************************************************************************************************************
```

```
     REPEAT_SEQUENCE


     REF
     PAGE ********************************************************************************************************************
          *                                                                                                                    *
263       *  1    .. PERFORM REPEAT TERMINATION LOGIC.                                                                        *
264       *  2    B(7) := B(7) - 1 ..DECREMENT REPEAT COUNTER.                                                                *
265       *  3    B(U(B)) := B(U(B)) + REPEAT_SY ..INCREMENT B REGISTER WITH SY FIELD OF REPEAT INSTRUCTION                   *
267       *  4    IF B(7) =0, THEN                                                                                            *
268       *  5        CLEAR REPEAT_IN_PROGRESS INDICATOR                                                                      *
269       *  6    ELSE ..CHECK REPEAT TERMINATION CONDITIONS (OTHER THAN B(7)=0).                                             *
270       *  7        IF (U(31,6)>=42 .AND. U(31,6)<50) .OR. (U(31,6)=0'03' .AND. U(22,3)=7) "TSF", THEN ..A                 /*
          *                 COMPARE INSTRUCTION.                                                                           *
272       *  8            DO CASE REPEAT_AB(4) OF                                                                         *
273       *  9                \0\ IF ASR(2,1)=0 "<>", THEN CLEAR REPEAT_IN_PROGRESS ENDIF                                 *
274       * 10                \1\ IF ASR(2,1)=1 "=", THEN CLEAR REPEAT_IN_PROGRESS ENDIF                                  *
275       * 11                \2\ IF ASR(2,2)=B'01' ">", THEN CLEAR REPEAT_IN_PROGRESS ENDIF                              *
276       * 12                \3\ IF ASR(1,1)=1 ">=", THEN CLEAR REPEAT_IN_PROGRESS ENDIF                                 *
277       * 13                \4\ IF ASR(1,1)=0 "<", THEN CLEAR REPEAT_IN_PROGRESS ENDIF                                  *
278       * 14                \5\ IF ASR(2,1)=1 .OR. ASR(1,1)=0 "<=", THEN CLEAR REPEAT_IN_PROGRESS ENDIF                 *
280       * 15                \6\ IF ASR(0,1)=1 "OUT OF LIMITS", THEN CLEAR REPEAT_IN_PROGRESS ENDIF                      *
282       * 16                \7\ IF ASR(0,1)=0 "WITHIN LIMITS", THEN CLEAR REPEAT_IN_PROGRESS ENDIF                      *
284       * 17            ENDDO ..END CASE.                                                                               *
285       * 18        ELSE ..NOT A COMPARE INSTRUCTION.                                                                   *
286       * 19            DO CASE REPEAT_AB(A) OF                                                                         *
287       * 20                \0\ IF REPEAT_ACCUMULATOR <>0, THEN CLEAR REPEAT_IN_PROGRESS ENDIF                          *
288       * 21                \1\ IF REPEAT_ACCUMULATOR =0, THEN CLEAR REPEAT_IN_PROGRESS ENDIF                           *
289       * 22                \2\ IF REPEAT_ACCUMULATOR >=0, THEN CLEAR REPEAT_IN_PROGRESS ENDIF                          *
290       * 23                \3\ IF REPEAT_ACCUMULATOR <0, THEN CLEAR REPEAT_IN_PROGRESS ENDIF                           *
291       * 24                ... CASE 4 IMPLIES DO NOT TERMINATE                                                         *
292       * 25                \5\ IF REPEAT_ACCUMULATOR EVEN PARITY .AND. MEMORY_STORE_INDICATOR, THEN CLEAR              /*
          *                       REPEAT_IN_PROGRESS ENDIF                                                                 *
294       * 26                \6\ IF REPEAT_ACCUMULATOR ODD PARITY .AND. MEMORY_STORE_INDICATOR, THEN CLEAR               /*
          *                       REPEAT_IN_PROGRESS ENDIF                                                                 *
296       * 27                ... CASE 7 IMPLIES DO NOT TERMINATE                                                         *
297       * 28            ENDDO ..END CASE.                                                                               *
298       * 29        ENDIF                                                                                               *
299       * 30    ENDIF                                                                                                   *
300       * 31    IF .NOT. REPEAT_IN_PROGRESS, THEN                                                                       *
301       * 32        CLEAR INTERRUPT_SCAN_INHIBIT INDICATOR                                                              *
302       * 33    ENDIF                                                                                                   *
303       * 34    RETURN                                                                                                  *
          *                                                                                                                    *
          ********************************************************************************************************************
```

****
* *
* CP SUPPORT ROUTINES LEVEL I *
* *
****

```
      GENERATE_SYNCHRONOUS_INTERRUPT (P_MODIFICATION, _INTERRUPT, _CODE)

      REF
      PAGE ********************************************************************************************************************
           *                                                                                                                    *
306        *  1    .. GENERATE THE APPROPRIATE SYNCHRONOUS INTERRUPT, BUILD THE INTERRUPT STATUS CODE (ISC),                   *
308        *  2    ..AND MODIFY PROGRAM COUNTER FOR STORAGE AS INDICATED.                                                      *
309        *  3    ..CURRENT INSTRUCTION IS ABORTED WHETHER OR NOT THE INTERRUPT IS ACTUALLY GENERATED.                        *
311        *  4    .. (1) P_MODIFICATION - P REGISTER MODIFICATION VALUE.                                                      *
312        *  5    .. (2) _INTERRUPT - INTERRUPT INFORMATION.                                                                  *
313        *  6    .. (3) _CODE - INFORMATION FOR ISC GENERATION (MEMORY BANK, IOC#, ETC.).                                    *
315        *  7    IF _INTERRUPT IS A CLASS I, THEN                                                                            *
316        *  8        CLASS := I ..CLASS INDICATOR FOR INTERRUPT SEQUENCE.                                                    *
317        *  9        IF ASR(14,1) "CLASS I LOCKED OUT", THEN                                                                 *
318        * 10            RETURN ..TO MAIN LOOP (LOCKED OUT IMPLIES NO INTERRUPT GENERATED).                                  *
319        * 11        ENDIF                                                                                                   *
320        * 12    ELSE                                                                                                        *
321        * 13        IF _INTERRUPT IS A CLASS II, THEN                                                                       *
322        * 14            CLASS := II ..CLASS INDICATOR FOR INTERRUPT SEQUENCE.                                               *
323        * 15            IF ASR(13,1) "CLASS II LOCKED OUT", THEN                                                            *
324        * 16                RETURN ..TO MAIN LOOP (LOCKED OUT IMPLIES NO INTERRUPT GENERATED).                              *
325        * 17            ENDIF                                                                                               *
326        * 18        ELSE ..MUST BE CLASS IV (REI) SINCE THERE ARE NO CLASS III SYNCHRONOUS INTERRUPTS.                      *
328        * 19            CLASS := IV ..CLASS INDICATOR FOR INTERRUPT SEQUENCE.                                               *
329        * 20            ..NOTE: THE CLASS IV INTERRUPT (REI) IS NEVER LOCKED OUT.                                           *
330        * 21        ENDIF                                                                                                   *
331        * 22    ENDIF                                                                                                       *
332        * 23    DO CASE _INTERRUPT OF                                                                                       *
333        * 24        \OP MEMORY RESUME\ ISC := _CODE .CON. B'0000' ..CODE = MEMORY BANK NUMBER (0-15).                       *
335        * 25        \IOC COMMAND RESUME\ ISC := _CODE .CON. B'00000001' ..CODE = IOC NUMBER.                                *
337        * 26        \INSTRUCTION MEMORY RESUME\ ISC := _CODE .CON. B'0010' ..CODE = MEMORY BANK NUMBER (0-15).              *
339        * 27        \IOC INTERRUPT CODE RESUME\ ISC := _CODE .CON. B'00000011' ..CODE = IOC NUMBER.                         *
341        * 28        \FLOATING POINT ERROR\ ISC := B'0001'                                                                   *
342        * 29        \CP ILLEGAL INSTRUCTION ERROR\ ISC := B'0010'                                                           *
343        * 30        \PRIVILEGED INSTRUCTION ERROR\ ISC := B'0011'                                                           *
344        * 31        \OPERAND BREAKPOINT MATCH\ ISC := B'0101'                                                               *
345        * 32        \OPERAND READ OR INDIRECT ADDRESSING\ ISC := B'0110'                                                    *
346        * 33        \OPERAND WRITE\ ISC := B'1001'                                                                          *
347        * 34        \OPERAND LIMIT\ ISC := B'1010'                                                                          *
348        * 35        \INSTRUCTION BREAKPOINT MATCH\ ISC := B'1011'                                                           *
349        * 36        \INSTRUCTION EXECUTE\ ISC := B'1101'                                                                    *
350        * 37        \INSTRUCTION LIMIT\ ISC := B'1110'                                                                      *
351        * 38        \REI\ ISC := _CODE ..CODE EQUIVALENT TO ISC FOR REI.                                                    *
352        * 39    ENDDO ..END CASE.                                                                                           *
353        * 40    DO CASE ..MODIFY PROGRAM COUNTER FOR INTERRUPT STORAGE.                                                     *
354        * 41        \REPEAT_PENDING\ P := P-1 ..REEXECUTE REPEAT.                                                           *
355        * 42        \REPEAT_IN_PROGRESS\ P := P-2 ..REEXECUTE REPEAT.                                                       *
356        * 43    ELSE                                                                                                        *
357        * 44        P := P-P_MODIFICATION ..SO PROPER RETURN CAN BE MADE.                                                   *
358        * 45    ENDDO ..END CASE.                                                                                           *
359      8 * 46    CALL INTERRUPT_SEQUENCE (CLASS, ISC)                                                                        *
360        * 47    RETURN                                                                                                      *
           *                                                                                                                    *
           **********************************************************************************************************************
```

```
      GET_ISC (_CLASS, REF _CODE)

      REF
      PAGE ***************************************************************************************************************
           *                                                                                                             *
362        *  1    .. ACQUIRE INTERRUPT STATUS CODE (ISC) FROM APPROPRIATE IOC AND ADD                                   *
363        *  2    ..IOC NUMBER IN BITS 9 & 8 BEFORE RETURNING (ISC) TO CALLER. SHOULD                                   *
364        *  3    ..THE IOC RETURN AN INVALID ISC, THIS FACT MUST BE CONVEYED TO THE CALLER.                            *
366        *  4    .. (1) _CLASS - INTERRUPT LEVEL (1 OR 3) OF CONCERN.                                                  *
367        *  5    .. (2) _CODE - ISC RECEIVED FROM IOC AND RETURNED TO CALLER.                                          *
360        *  6    DETERMINE IOC_# CAUSING INTERRUPT                                                                     *
369        *  7    RECOGNIZE RECEIPT OF INTERRUPT BY MAKING AN IOC REQUEST ON IOC_#                                      *
370        *  8    SEND INTERRUPT CLASS DESIGNATOR (_CLASS) VIA O_BUS                                                    *
371        *  9    WAIT UNTIL IOC ACQUIRES CLASS INDICATOR                                                               *
372        * 10    RECEIVE ISC FROM IOC_# ACROSS O_BUS                                                                   *
373        * 11    REACTIVATE CLASS(_CLASS) INTERRUPT LINE                                                               *
374        * 12    _CODE := IOC_# .CON. ISC ..2-BIT IOC # CONCATENATED WITH 8-BIT ISC (POSSIBLY INVALID).               *
376        * 13    RETURN                                                                                                *
           *                                                                                                             *
           ***************************************************************************************************************
```

```
     JUMP_ADDRESS(M_DISPLACEMENT, REF OPERAND_S, REF OPERAND_DISPLACEMENT)

     REF
     PAGE ********************************************************************************************************************
          *                                                                                                                    *
378       *  1    .. CALLED BY JUMP TYPE INSTRUCTIONS. FINDS THE "JUMP TO" ADDRESS AND                                        *
379       *  2    ..PERFORMS BREAKPOINT AND SPR CHECKS.                                                                       *
380       *  3    .. (1) M_DISPLACEMENT - 0/1 FOR OP1/OP2 SEQUENCE RESPECTIVELY.                                              *
381       *  4    .. (2) OPERAND_S - THE BASE REGISTER PORTION OF THE ADDRESS.                                                 *
382       *  5    .. (3) OPERAND_DISPLACEMENT - THE 16-BIT DISPACEMENT PORTION OF THE ADDRESS.                                 *
384       *  6    IF U(I) "INDIRECT ADDRESSING", THEN ..DO CASCADING.                                                         *
385    18 *  7        CALL IA_SEQUENCE (OPERAND_S, OPERAND_DISPLACEMENT, DUM_C, DUM_C1, DUM_P,DUM_MASK)                       *
387       *  8        ..DUMMY C, P, MASK SINCE CHARACTER ADDRESSING HAS NO MEANING FOR JUMP INSTRUCTIONS.                     *
389       *  9        OPERAND_DISPLACEMENT := OPERAND_DISPLACEMENT + M_DISPLACEMENT                                           *
390       * 10    ELSE                                                                                                        *
391       * 11        OPERAND_DISPLACEMENT := U(Y) + B(U(B))(15) + M_DISPLACEMENT                                             *
392       * 12        OPERAND_S := U(S)                                                                                       *
393       * 13    ENDIF                                                                                                       *
394       * 14    32_REG(1) := OPERAND_DISPLACEMENT ..GET COPY FOR USE IN CHECKING.                                           *
395    20 * 15    CALL SPR_CHECK (OPERAND_S, INSTRUCTION EXECUTE, "P-"1, OPERAND_DISPLACEMENT) ..ECP(66).                     *
397    32 * 16    CALL ADD_S (OPERAND_S, 32_REG(1)) ..COMPUTE FINAL (ABSOLUTE) ADDRESS.                                       *
398    19 * 17    CALL MEMORY_READ (32_REG(1), DUMMY, OPERAND_S, "P-"1) ..CHECK FOR OPERAND MEMORY RESUME.                    *
400    17 * 18    CALL BPR_CHECK (32_REG(1), OPERAND) ..CHECK FOR OPERAND BREAKPOINT.                                         *
401       * 19    RETURN                                                                                                      *
          *                                                                                                                    *
          ********************************************************************************************************************
```

```
     OP_READ (REF 32_REG, M_DISPLACEMENT)

     REF
     PAGE ********************************************************************************************************************
          *                                                                                                                    *
403       *  1    .. DETERMINE OPERAND ADDRESS FROM U REGISTER USING I, K, S, Y & B INDICATORS.                               *
405       *  2    ..TRANSFER THE 32-BIT OPERAND (AS DETERMINED BY OPERAND TYPE) TO THE REGISTER SUPPLIED BY CALLER.           *
407       *  3    .. (1) 32_REG - 32-BIT REFERENCE REGISTER.                                                                  *
408       *  4    .. (2) M_DISPLACEMENT - 0/1 FOR OP1/OP2 SEQUENCE RESPECTIVELY.                                              *
409       *  5    IF U(I) "INDIRECT ADDRESSING", THEN                                                                         *
410    18 *  6        CALL IA_SEQUENCE(S_DESIGNATOR, Y, _C, _C1, _P, _MASK) ..FIND FINAL OPERAND ADDRESS VALUES.              *
412       *  7        OPERAND_DISPLACEMENT := OPERAND_DISPLACEMENT + M_DISPLACEMENT                                           *
413       *  8    ELSE ..COMPUTE FINAL OPERAND ADDRESS VALUES.                                                                *
414       *  9        Y := U(Y)+B(U(B))(15) ..ADD DISPLACEMENT AND INDEX REGISTER.                                            *
415       * 10        Y := Y+M_DISPLACEMENT ..ADD IN M_DISPLACEMENT FOR DESIRED OP1 OR OP2 SEQUENCE.                          *
417       * 11        S_DESIGNATOR := U(S)                                                                                    *
418       * 12        IF CHARACTER_ADDRESSING_OVERRIDE, THEN                                                                  *
419       * 13            ACQUIRE ACTIVE _C, _P, AND _MASK                                                                    *
420       * 14        ELSE                                                                                                    *
421       * 15            _C := B'10' ..FORCE FOLLOWING CODE TO USE K DESIGNATOR.                                             *
422       * 16        ENDIE                                                                                                   *
423       * 17    ENDIE                                                                                                       *
424       * 18    IF .NOT. (INSTRUCTION_FORMAT_INDICATOR=I .AND. K=0 .AND. _C EVEN) "NOT IMMEDIATE OPERAND", THEN             *
426    20 * 19        CALL SPR_CHECK (S_DESIGNATOR, OPERAND READ, "P-"1)                                                      *
427    32 * 20        CALL ADD_S (S_DESIGNATOR, Y)                                                                            *
428    17 * 21        CALL BPR_CHECK (Y, OPERAND)                                                                             *
429    19 * 22        CALL MEMORY_READ (Y, 32_REG, S_DESIGNATOR, OPERAND, "P-"1)                                              *
430       * 23    ELSE "HANDLE IMMEDIATE OPERAND"                                                                             *
431       * 24        32_REG := U(SY)                                                                                         *
432       * 25        IF _C <> 0 .OR. _C1 <> 0, THEN ..IMMEDIATE OPERAND FOR IWS INDIRECT ADDRESS DOESN'T USE                 /*
                            B(U(B)).                                                                                         *
434       * 26            32_REG := 32_REG + B(U(B))                                                                          *
435       * 27        ENDIE                                                                                                   *
436       * 28    ENDIE                                                                                                       *
437       * 29    IF _C EVEN, THEN ..NOT CHARACTER ADDRESSING.                                                                *
438       * 30        IF INSTRUCTION_FORMAT_INDICATOR=I, THEN                                                                 *
439       * 31            ADJUST 32_REG ACCORDING TO U(K) ..INCLUDES IMMEDIATE OPERANDS.                                      *
440       * 32        ENDIE                                                                                                   *
441       * 33    ELSE ..CHARACTER ADDRESSING.                                                                                *
442       * 34        RIGHT JUSTIFY / ZERO FILL 32_REG.                                                                       *
443       * 35        ..THIS CAN BE DONE BY SHIFTING 32_REG RIGHT LOGICAL _P BITS & ANDING WITH _MASK.                        *
445       * 36    ENDIE                                                                                                       *
446       * 37    RETURN                                                                                                      *
          *                                                                                                                    *
          ********************************************************************************************************************
```

```
     OP_STORE (32_REG, M_DISPLACEMENT)

     REF
     PAGE ********************************************************************************************************************
          *                                                                                                                    *
448       *  1    .. DETERMINE OPERAND ADDRESS FROM THE U REGISTER USING I, K, S, Y & B INDICATORS.                           *
450       *  2    ..STORE THE QUANTITY SUPPLIED BY CALLER IN MEMORY.                                                          *
451       *  3    .. (1) 32_REG - 32-BIT VALUE TO TRANSFER INTO MEMORY SUBJECT TO OPERAND TYPE RESTRICTIONS.                  *
453       *  4    .. (2) M_DISPLACEMENT - 0/1 FOR OP1/OP2 SEQUENCE RESPECTIVELY.                                             *
454       *  5    IF U(I) "INDIRECT ADDRESSING", THEN                                                                         *
455    18 *  6        CALL IA_SEQUENCE(S_DESIGNATOR, Y, _C, _C1, _P, _MASK) ..FIND FINAL OPERAND ADDRESS VALUES.              *
457       *  7        OPERAND_DISPLACEMENT := OPERAND_DISPLACEMENT + M_DISPLACEMENT                                           *
458       *  8    ELSE ..COMPUTE FINAL OPERAND ADDRESS VALUES.                                                                *
459       *  9        Y := U(Y)+B(U(B))(15) ..ADD DISPLACEMENT AND INDEX REGISTER.                                            *
460       * 10        Y := Y+M_DISPLACEMENT ..ADD IN M_DISPLACEMENT FOR DESIRED OP1 OR OP2 SEQUENCE.                          *
462       * 11        S_DESIGNATOR := U(S)                                                                                    *
463       * 12        IF CHARACTER_ADDRESSING_OVERRIDE, THEN                                                                  *
464       * 13            ACQUIRE ACTIVE _C, _P AND _MASK                                                                     *
465       * 14        ELSE                                                                                                    *
466       * 15            _C := B'10' ..FORCE FOLLOWING CODE TO USE K DESIGNATOR.                                             *
467       * 16        ENDIF                                                                                                   *
468       * 17    ENDIF                                                                                                       *
469       * 18    SET MEMORY_STORE_INDICATOR ..FOR USE IN REPEAT TERMINATION DETECTION.                                       *
470       * 19    IF NOT NOPO REFERENCE, THEN                                                                                 *
471       * 20        IF .NOT.(INSTRUCTION_FORMAT_INDICATOR=I .AND. K=0 .AND. _C EVEN) "NOT IMMEDIATE OPERAND",               /*
          *              THEN                                                                                                *
473    20 * 21            CALL SPR_CHECK (S_DESIGNATOR, OPERAND WRITE, "P-"1, Y)                                           *
474    32 * 22            CALL ADD_S (S_DESIGNATOR, Y)                                                                     *
475    17 * 23            CALL BPR_CHECK (Y, OPERAND)                                                                      *
476    19 * 24            CALL MEMORY_READ (Y, VALUE, S_DESIGNATOR, OPERAND, "P-"1)                                        *
477       * 25            IF _C EVEN, THEN ..NOT CHARACTER ADDRESSING                                                      *
478       * 26                IF INSTRUCTION_FORMAT_INDICATOR = I, THEN                                                    *
479       * 27                    ADJUST 32_REG ACCORDING TO U(K) AND MASK INTO VALUE                                      *
480       * 28                ENDIF                                                                                        *
481       * 29            ELSE ..CHARACTER ADDRESSING                                                                      *
482       * 30                POSITION CHARACTER AND MASK INTO VALUE                                                       *
483       * 31            ENDIF                                                                                            *
484       * 32            TRANSFER VALUE TO MEMORY(Y) ..STORE TO NOPO LOCATION IS A NOOP.                                  *
485       * 33        ENDIF ..FORMAT I INSTRUCTION WITH K=0 IS A NOOP                                                      *
486       * 34    ENDIF                                                                                                       *
487       * 35    RETURN                                                                                                      *
          *                                                                                                                    *
          ********************************************************************************************************************
```

```
**********************************
*                                *
* CP SUPPORT ROUTINES LEVEL II   *
*                                *
**********************************
```

```
BPR_CHECK (ABS_ADDR, TYPE_)

REF
PAGE ************************************************************************************************************************
     *                                                                                                                      *
490  *  1    .. PERFORM BREAKPOINT REGISTER CHECKS. THE UYK7 EMULATION HAS EIGHT                                             *
491  *  2    ..BREAKPOINT REGISTERS. ONE REGISTER CORRESPONDS TO THE ACTUAL HARDWARE                                         *
492  *  3    ..BREAKPOINT REGISTER OF THE REAL UYK-7 MACHINE. THE OTHER SEVEN                                                *
493  *  4    ..REGISTERS ARE PSEUDO BREAKPOINT REGISTERS AND ARE CONSIDERED                                                  *
494  *  5    ..EXTENTIONS OF THE UYK-7 ARCHITECTURE. THE PSEUDO BREAKPOINT REGISTERS                                         *
495  *  6    ..ARE LOCATED IN CPP LOCATIONS 61-67 (PSEUDO BREAKPOINTS ARE NOT                                                *
496  *  7    ..ACCESSIBLE TO UYK-7 PROGRAMS). IF PSEUDO_BREAKPOINTS ARE ENABLED                                              *
497  *  8    ..ALL BREAKPOINT REGISTERS WILL BE CHECKED (TOTAL OF 8), OTHERWISE, ONLY                                        *
498  *  9    ..THE EMULATED HARDWARE REGISTER WILL BE CHECKED. TYPE_ WILL BE TESTED                                          *
499  * 10    ..AGAINST THE TYPE FIELD OF THE ACTIVE BREAKPOINT REGISTERS AND IF THEY                                         *
500  * 11    ..MATCH, ABS_ADDR WILL BE COMPARED TO THE BREAKPOINT REGISTER COMPARISON                                        *
501  * 12    ..ADDRESS(BITS 0-17). FOR THE EMULATED HARDWARE REGISTER, A 'HIT' ON                                           *
502  * 13    ..ADDRESS COMPARISON CAUSES A 'HALT' OR CLASS II INTERRUPT DEPENDING ON                                        *
503  * 14    ..THE PROGRAM/MANUAL SWITCH. FOR THE PSEUDO BREAKPOINT REGISTERS, A                                            *
504  * 15    ..'HIT' ON ADDRESS COMPARISON CAUSES A 'HALT' REGARDLESS OF THE                                                *
505  * 16    ..PROGRAM/MANUAL SWITCH SETTING.                                                                               *
506  * 17    ..(1) ABS_ADDR - 18 BIT ABSOLUTE UYK-7 ADDRESS                                                                 *
507  * 18    ..(2) TYPE_ - TYPE OF BREAKPOINT CHECK DESIRED                                                                 *
508  * 19    IF PSEUDO_BREAKPOINTS ENABLED, THEN                                                                            *
509  * 20        CNT := TOTAL # OF PSEUDO BREAKPOINTS ..CHECK ALL BREAKPOINT REGISTERS.                                     *
510  * 21    ELSE                                                                                                           *
511  * 22        CNT := 0 ..ONLY EMULATED HARDWARE BREAKPOINT CHECKED.                                                      *
512  * 23    ENDIF                                                                                                          *
513  * 24    DO WHILE CNT >= 0 ..CYCLE THRU ACTIVE BREAKPOINT REGISTERS.                                                    *
514  * 25        IF CHR(BREAKPOINT_REGISTER+CNT)(19,2) <> 0, THEN                                                           *
515  * 26            IF (TYPE_=CHR(BREAKPOINT_REGISTER+CNT)(19,2) .OR.                                                      *
516  * 27                CHR(BREAKPOINT_REGISTER+CNT)(19,2)=3) "MATCHING TYPES", THEN                                       *
517  * 28                IF ABS_ADDR = CHR(BREAKPOINT_REGISTER+CNT)(17) "MATCHING ADDRESS", THEN                            *
518  * 29                    IF MANUAL MODE .OR. CNT >0 "PSEUDO BKPT.", THEN                                                *
519  * 30                        SET _STEP ..STOP IN MAIN LOOP IF OPERAND BREAKPOINT.                                       *
520  * 31                        IF TYPE_ = INSTRUCTION BREAKPOINT, THEN                                                    *
521  * 32                            INTERRUPT INSTRUCTION EXECUTION..STOP IMMEDIATELY IF INSTRUCTION TYPE.                 *
522  * 33                        ENDIF                                                                                      *
523  * 34                    ELSE                                                                                           *
524  * 35                        IF ASR(13,1) "CLASS II LOCKED OUT", THEN                                                   *
525  * 36                            RETURN ..BREAKPOINT INTERRUPTS DISABLED.                                               *
526  * 37                        ELSE                                                                                       *
527  * 38                            IF TYPE_ = INSTRUCTION BREAKPOINT, THEN                                                *
528  11 * 39                            GENERATE_SYNCHRONOUS_INTERRUPT ("P-"0, INSTRUCTION BREAKPOINT)                     *
529  * 40                                ..ABORT INSTRUCTION.                                                               *
530  * 41                            ELSE                                                                                   *
531  11 * 42                            GENERATE_SYNCHRONOUS_INTERRUPT ("P-"1, OPERAND BREAKPOINT)                         *
532  * 43                                .. ABORT INSTRUCTION.                                                              *
533  * 44                            ENDIF                                                                                  *
534  * 45                        ENDIF                                                                                      *
535  * 46                    ENDIF                                                                                          *
536  * 47                ENDIF                                                                                              *
537  * 48            ENDIF                                                                                                  *
538  * 49        ENDIF                                                                                                      *
539  * 50        CNT := CNT-1                                                                                               *
540  * 51    ENDDO ..END WHILE.                                                                                             *
541  * 52    RETURN                                                                                                         *
     *                                                                                                                      *
```

```
     IA_SEQUENCE (REF S_DESIGNATOR,REF Y,REF _C,REF _C1,REF _P,REF _MASK)

REF
PAGE ********************************************************************************************************************
     *                                                                                                                    *
543  *  1    .. PERFORMS INDIRECT ADDRESSING RETURNING THE FOLLOWING VALUES:                                              *
544  *  2    .. (1) S_DESIGNATOR - S DESIGNATOR OF OPERAND.                                                               *
545  *  3    .. (2) Y - DISPLACEMENT PORTION OF THE OPERAND ADDRESS.                                                      *
546  *  4    .. (3) _C, _P & _MASK - CHARACTER ADDRESSING PARAMETERS WHEN NEEDED.                                         *
547  *  5    .. CALLER IS RESPONSIBLE FOR ENSURING THAT AN IA_SEQUENCE IS REQUIRED.                                       *
548  *  6    Y := U(Y) + B(U(B))(15)                                                                                      *
549  *  7    S_DESIGNATOR := U(S)                                                                                         *
550  *  8    DO WHILE U(I) "INDIRECTION" ..CASCADE IGNORING U(X).                                                         *
551  20 *  9        CALL SPB_CHECK (S_DESIGNATOR,INDIRECT ADDRESSING,"9-"1,Y)                                            *
552  *  10       32_REG(1) := Y  .. SAVE Y FOR POSSIBLE COMPARISON TO SPB.                                                *
553  32 *  11       CALL ADD_S (S_DESIGNATOR, 32_REG(1))                                                                 *
554  17 *  12       CALL BPB_CHECK(32_REG(1),OPERAND)                                                                    *
555  19 *  13       CALL MEMORY_READ(32_REG(1),IACW,S_DESIGNATOR,OPERAND,"9-"1) ..GET INDIRECT ADDRESS CONTROL           *
     *                 WORD (IACW).                                                                                     *
557  *  14       U(19,20) := IACW(19,20) ..UPDATE THE U REGISTER B, I, S & Y FIELDS.                                      *
558  *  15       .. EXPERIMENTS INDICATE THAT U(19,17) IS ALWAYS SET TO 0 FOR THE LAST IACW IF THAT IACW IS OF            *
560  *  16       ..THE SPECIAL BASE (IWS) TYPE. SINCE NO DOCUMENTATION CAN BE FOUND TO SUPPORT THIS                       *
562  *  17       ..OBSERVATION IT IS OMITTED FROM THIS DESIGN.                                                            *
563  *  18       DO CASE IACW(31,3) ..FORM NEW Y AND S_DESIGNATOR ACCORDING TO IACW.                                      *
564  *  19           .. \B"000"\ Y:= U(SY)                                                                                *
565  *  20           .. S_DESIGNATOR := U(B)                                                                              *
566  *  21           .. \3"001"\ Y:= U(SY) + B(U(B))(15)                                                                  *
567  *  22           .. S_DESIGNATOR := B(U(B))(19,3)                                                                     *
568  *  23           .. ELSE Y:= U(Y) + B(U(B))(15)                                                                       *
569  *  24           .. S_DESIGNATOR := U(S)                                                                              *
570  *  25       ENDDO ..END CASE                                                                                         *
571  *  26   ENDDO ..END WHILE                                                                                           *
572  *  27   _C := IACW(C) ..RETURN DATA STRUCTURE INFORMATION TO CALLER.                                                 *
573  *  28   _C1 := IACW(C1) ..I.E. SAVE BIT 29 IN CASE X=0, FMT I, & IWS OR IWB.                                        *
574  *  29   IF _C = 00 .AND. REPEAT_IN_PROGRESS "REPEAT OF SPECIAL INDIRECT", THEN                                       *
575  11 *  30       CALL GENERATE_SYNCHRONOUS_INTERRUPT ("9-"2,CP ILLEGAL INSTRUCTION) ..ABORT THE INSTRUCTION.          *
577  *  31   ENDIF                                                                                                       *
578  *  32   IF _C = 00 .AND. U(F)=0"25" "SXB ILLEGAL IF I=1 AND C=00", THEN                                             *
579  11 *  33       CALL GENERATE_SYNCHRONOUS_INTERRUPT ("9-"1,CP ILLEGAL INSTRUCTION) ..ABORT THE INSTRUCTION.          *
581  *  34   ENDIF                                                                                                       *
582  *  35   IF _C 000 "CHARACTER REFERENCE", THEN                                                                      *
583  *  36       _P := IACW(P)                                                                                           *
584  *  37       _MASK := IACW(W) 1 BITS RIGHT JUSTIFIED IN A 32-BIT WORD                                                *
585  *  38       SET CHARACTER_ADDRESSING_OVERRIDE                                                                       *
586  *  39       IF SEQUENTIAL CHARACTER ADDRESSING .AND. NOT WORD REFERENCE, THEN                                       *
587  *  40           .. MODIFY IACW(P) AND MAYBE IACW(Y).                                                                *
588  *  41           IF (P-IACW(W)) < 0, THEN .. MUST UPDATE IACW(Y)                                                     *
589  *  42               IACW(P) := 32 - IACW(W); IACW(Y) := IACW(Y) + 1                                                  *
590  *  43           ELSE                                                                                                *
591  *  44               IACW(P) := IACW(P) - IACW(W)                                                                     *
592  *  45           ENDIF                                                                                               *
593  *  46       ENDIF                                                                                                   *
594  *  47   ENDIF                                                                                                       *
595  *  48   RETURN                                                                                                      *
     *                                                                                                                    *
     ********************************************************************************************************************
```

```
NSWC          AN/UYK-7 (CP)
              CP SUPPORT ROUTINES LEVEL II

     MEMORY_READ (.16_REG, REF 32_REG, S_DESIGNATOR, RESUME_TYPE, P_MOD)

     REF
     PAGE ****************************************************************************************************
          *
          *  1    .. FETCH A 32-BIT QUANTITY FROM MEMORY(NORO OR MAIN) INTO THE REGISTER SUPPLIED BY THE CALLER.
          *  2    .. (1) 16_REG - 16-BIT ABSOLUTE ADDRESS TO READ FROM.
          *  3    .. (2) 32_REG - 32-BIT REFERENCE REGISTER.
          *  4    .. (3) S_DESIGNATOR - BASE REGISTER ASSOCIATED WITH THE ADDRESS.
          *  5    .. (4) RESUME_TYPE - DESIRED MEMORY RESUME ERROR IF ADDRESS IS OUT OF BOUNDS.
          *  6    .. (5) P_MOD - P REGISTER MODIFICATION IF AN ERROR OCCURS.
          *  7    IF (S_DESIGNATOR=7 .AND. ASR(7,1) .AND. 16_REG<512) "NORO REFERENCE", THEN
          *  8        32_REG := _NORO(16_REG)
          *  9    ELSE ..FETCH FROM MAIN_MEMORY.
          * 10        IF 16_REG EXCEEDS MEMORY LIMIT, THEN
    11    * 11            GENERATE_SYNCHRONOUS_INTERRUPT (P_MOD, RESUME_TYPE, 16_REG/16K)
          * 12            ..ABORT CURRENT INSTRUCTION.
          * 13        ELSE
          * 14            32_REG := MAIN_MEMORY(16_REG)
          * 15        ENDIF
          * 16    ENDIF
          * 17    RETURN
          *
          ****************************************************************************************************
```

```
SPR_CHECK (S_INDICATOR, CHECK_TYPE, P_MODIFICATION, 16_REG)

REF
PAGE ********************************************************************************************************************
     *                                                                                                                  *
618  *  1    .. IF PROTECTION CHECKING IS ENABLED (AS DETERMINED IN MAIN LOOP),                                          *
619  *  2    ..THIS ROUTINE PERFORMS THE ACCESS TYPE CHECK                                                               *
620  *  3    ..DESIGNATED BY CALLER AND THE DISPLACEMENT CHECK. ON                                                       *
621  *  4    ..ANY PROTECTION VIOLATION P IS MODIFIED AS INDICATED, THE                                                  *
622  *  5    ..APPROPRIATE INTERRUPT GENERATED, AND THE INSTRUCTION                                                      *
623  *  6    ..IS ABORTED. SOFTWARE PROTECTION REGISTER (SPR) FIELDS (E.G. IA, R,                                        *
624  *  7    ..IR, ETC.) ARE AS DEFINED ON THE REPERTOIRE CARD.                                                          *
625  *  8    .. (1) S_INDICATOR - BASE REGISTER ASSOCIATED WITH THE ADDRESS                                              *
626  *  9    .. (2) CHECK_TYPE - SPECIFIES WHETHER OPERAND, INSTRUCTION OR INDIRECT ADDRESSING TYPE                      *
628  * 10    .. (3) P_MODIFICATION - P REGISTER MODIFICATION IF AN ERROR OCCURS                                          *
629  * 11    .. (4) 16_REG - VALUE TO BE COMPARED AGAINST SPR LIMIT FIELD                                                *
630  * 12    IF SPR_CHECKS ENABLED, THEN                                                                                 *
631  * 13        FETCH APPROPRIATE SPR ..(I.E. CHR(0*160**+S_INDICATOR))                                                 *
632  * 14        IF CHECK_TYPE VALID FOR THIS SPR, THEN                                                                  *
633  * 15            IF (CHECK_TYPE = IA "INDIRECT ADDRESSING, THEN                                                      *
634  * 16                IF SPR_PRIVILEGED_INSTRUCTION SET .AND. SPR(IR), THEN                                           *
635  * 17                    ..SPECIAL PRIVILEGED INSTRUCTION(IT ON REP CARD) => INST(I) SET AND SPR(IR) SET             *
637  * 18                    CALL GENERATE_SYNCHRONOUS_INTERRUPT(P_MODIFICATION, PRIVILEGED INSTRUCTION)                 *
639  * 19                    ..ABORT THE INSTRUCTION.                                                                    *
640  * 20                ELSE                                                                                            *
641  * 21                    IF ASR(11,2) .AND.SPR(IR) = 0, THEN                                                         *
642  * 22                        ..PROTECTION VIOLATION, ATTEMPT TO USE INTERRUPT B & S REGISTERS                        *
643  * 23                        ..FOR INDIRECT REFERENCE WITHOUT AUTHORIZATION.                                         *
644  * 24                        CALL GENERATE_SYNCHRONOUS_INTERRUPT (P_MODIFICATION, IA OR OP READ)                     *
645  * 25                        ..ABORT THE INSTRUCTION.                                                                *
646  * 26                    ENDIF                                                                                       *
647  * 27                ENDIF                                                                                           *
648  * 28            ENDIF                                                                                               *
649  * 29            IF 16_REG>SPR(R), THEN                                                                              *
650  * 30                GENERATE_SYNCHRONOUS_INTERRUPT ("P-" P_MODIFICATION, "APPROPRIATE LIMIT INTERRUPT")             *
652  * 31                ..ABORT CURRENT INSTRUCTION.                                                                    *
653  * 32            ENDIF                                                                                               *
654  * 33        ELSE ..THERE IS A PROTECTION VIOLATION.                                                                 *
655  * 34            GENERATE_SYNCHRONOUS_INTERRUPT ("P-" P_MODIFICATION, "APPROPRIATE TYPE VIOLATION")                  *
657  * 35            ..ABORT CURRENT INSTRUCTION.                                                                        *
658  * 36        ENDIF                                                                                                   *
659  * 37    ENDIF                                                                                                       *
660  * 38    RETURN                                                                                                      *
     *                                                                                                                  *
     ********************************************************************************************************************
```

CP INSTRUCTION SUPPORT ROUTINES

CP/IOC_CLOCK_COMMUNICATIONS

```
REF
PAGE ********************************************************************************************************************
     *
663  *  1    .. THIS ROUTINE CAN BE USED BY THOSE CP INSTRUCTIONS REQUIRING A                                            *
664  *  2    ..RESPONSE (VIA THE O BUS) FROM THE IOC. AFTER INITIATING A REQUEST                                         *
665  *  3    ..TO THE IOC, THE FUNCTION DESIGNATORS ARE SENT TO THE IOC ACCROSS THE                                      *
666  *  4    ..O BUS. THE IOC THEN RESPONDS WITH 32 BITS ON THE O BUS. THIS RESPONSE                                     *
667  *  5    ..WILL BE AVAILABLE TO THE CALLING INSTRUCTION.                                                             *
668  *  6    INITIATE A REQUEST TO THE IOC DESIGNATED BY U(A)                                                            *
669  *  7    .. ENSURE THAT THIS REQUEST HAS BEEN ACCEPTED.                                                              *
670  *  8    O_BUS(31,6) := U(F) ..SEND FUNCTION DESIGNATORS FROM U.                                                     *
671  *  9    O_BUS(22,3) := U(F2) ..TO INCLUDE SUBFUNCTION DESIGNATORS.                                                  *
672  * 10    O_BUS(19,1) := 0 ..ENSURE THAT THIS REQUEST NOT INTERPRETED AS AN INTTERRUPT STATUS CODE REQUEST.           *
674  * 11    SEND O_BUS TO THE IOC                                                                                       *
675  * 12    WAIT UNTIL IOC HAS ACQUIRED THE INFORMATION FROM THE O_BUS                                                  *
676  * 13    RECEIVE THE IOC'S RESPONSE ON THE O_BUS                                                                     *
677  * 14    RETURN ..INFORMATION NOW AVAILABLE ON O BUS.                                                                *
     *                                                                                                                 *
     *******************************************************************************************************************
```

```
      DO_JUMP (S_DESIGNATOR, _DISPLACEMENT)

      REF
      PAGE ********************************************************************************************************************
           *                                                                                                                    *
679        *  1    .. ROUTINE USED BY JUMP TYPE INSTRUCTIONS TO UPDATE THE PROGRAM COUNTER TO THE ADDRESS                       *
681        *  2    ..OF THE 'JUMPED TO' INSTRUCTION. CALLED WHEN A JUMP IS TO BE 'TAKEN'.                                       *
682        *  3    .. (1) S_DESIGNATOR - BASE REGISTER INDICATOR FOR NEW PROGRAM COUNTER (P(S)).                                *
684        *  4    .. (2) _DISPLACEMENT - DISPLACEMENT FOR NEW PROGRAM COUNTER (P(D)).                                          *
685        *  5    P(S) := S_DESIGNATOR                                                                                         *
686        *  6    P(D) := _DISPLACEMENT                                                                                        *
687        *  7    RETURN                                                                                                       *
           *                                                                                                                    *
           ********************************************************************************************************************
```

```
      GET_SHIFT_AMOUNT (_M, REF SHIFT_COUNT)

REF
PAGE ********************************************************************************************************************
     *                                                                                                                  *
689  *  1    .. RETURN SHIFT COUNT INTERPRETED FROM M FIELD OF FMT IV B INSTRUCTION.                                     *
690  *  2    .. (1) _M - M FIELD OF FMT IV B INSTRUCTION                                                                 *
691  *  3    .. (2) SHIFT_COUNT - SHIFT COUNT RETURNED TO CALLER                                                         *
692  *  4    IF _M(6,1) = 0, THEN                                                                                        *
693  *  5        SHIFT_COUNT := _M(5)                                                                                    *
694  *  6    ELSE                                                                                                        *
695  *  7        IF _M(5,1) = 0, THEN                                                                                    *
696  34 *  8            CALL GET_BREG (_M(3,3), 32_REG(1))                                                              *
697  *  9            SHIFT_COUNT := 32_REG(1)(5 ..6 BITS OF COUNT FOR DOUBLE SHIFTS.                                     *
698  * 10        ELSE                                                                                                    *
699  33 * 11            CALL GET_AREG (_M(3,3), 32_REG(1))                                                              *
700  * 12            SHIFT_COUNT := 32_REG(1)(5) ..6 BITS OF COUNT FOR DOUBLE SHIFTS.                                    *
701  * 13        ENDIF                                                                                                   *
702  * 14    ENDIF                                                                                                       *
703  * 15    RETURN                                                                                                      *
     *                                                                                                                  *
     ********************************************************************************************************************
```

NSWC AN/UYK-7 (CP)
CP INSTRUCTION SUPPORT ROUTINES

HALF-WORD_TOGGLE

```
REF
PAGE ********************************************************************************************************************
     *                                                                                                                    *
     *  1    .. UPDATE UPPER/LOWER DESIGNATOR (ASR(15,1)) UPON COMPLETION OF A HALF-WORD INSTRUCTION. CALLED              *
     *  2    ..BY ALL HALF-WORD INSTRUCTIONS AFTER CHECKING FOR SYNCHRONOUS INTERRUPTS. THIS ALLOWS                        *
     *  3    ..THE RIGHT ASR TO BE STOPED UPON INTERRUPT (ECP-34).                                                         *
     *  4    ..NOTE: A HALF-WORD INSTRUCTION EXECUTED REMOTELY DOES NOT AFFECT THE UPPER/LOWER DESIGNATOR.                 *
     *  5    IF .NOT. EXECUTE_REMOTE_IN_PROGRESS, THEN                                                                     *
     *  6        IF ASR(15,1), THEN                                                                                        *
     *  7            ASR(15,1) := 0 ..NEXT INSTRUCTION FROM UPPER HALF.                                                    *
     *  8        ELSE                                                                                                      *
     *  9            IF UL(F) >= 0'60' "ASSUME 2 HALF-WORD INSTRUCTIONS", THEN                                             *
     * 10                ASR(15,1) := 1 ..EXECUTE LOWER HALF NEXT.                                                         *
     * 11            ENDIF                                                                                                 *
     * 12        ENDIF                                                                                                     *
     * 13    ENDIF                                                                                                         *
     * 14    RETURN                                                                                                        *
     *                                                                                                                    *
     ********************************************************************************************************************
```

```
      REPLACE_CHECK (32_REG)

      REF
      PAGE ********************************************************************************************************************
           *                                                                                                                    *
723        *  1    .. IF THE CURRENT INSTRUCTION IS A REPLACE TYPE, THEN UPDATE THE MEMORY                                     *
724        *  2    ..LOCATION (Y) SPECIFIED BY THE OPERAND ADDRESS.                                                            *
725        *  3    .. (1) 32_REG - 32 BIT VALUE TO BE STORED IN MEMORY                                                         *
726        *  4    IF U(F) >= 03, THEN ..THIS IS A REPLACE TYPE INSTRUCTION.                                                   *
727     27 *  5        CALL _REPLACE (32_REG)                                                                                  *
728        *  6    ENDIF                                                                                                       *
729        *  7    RETURN                                                                                                      *
           *                                                                                                                    *
           ********************************************************************************************************************
```

```
     _REPLACE (32_REG)

     REF
     PAGE ***************************************************************************************************************
          *                                                                                                             *
731       *  1    .. STORE 32_REG INTO THE MEMORY LOCATION (Y) SPECIFIED BY THE OPERAND                                  *
732       *  2    ..ADDRESS. THE ALGORITHM USED TO COMPUTE THIS ADDRESS DIFFERS WHEN A                                   *
733       *  3    ..REPEAT IS IN PROGRESS.                                                                              *
734       *  4    .. (1) 32_REG - 32 BIT VALUE TO BE STORED IN MEMORY                                                   *
735       *  5    IF REPEAT_IN_PROGRESS .AND. REPEAT_AB(8) <> 0, THEN                                                   *
736       *  6        SAVE U(S)                                                                                         *
737       *  7        U(S) := 6 ..FORCE S6 ON STORE FOR REPEATED INSTRUCTIONS.                                          *
738    15 *  8        CALL OP_STORE (32_REG, "DISPLACE"0) ..32_REG -> MEMORY                                            *
739       *  9        RESTORE U(S)                                                                                      *
740       * 10    ELSE                                                                                                  *
741    15 * 11        CALL OP_STORE (32_REG, "DISPLACE"0) .. 32_REG -> MEMORY                                           *
742       * 12    ENDIF                                                                                                 *
743       * 13    RETURN                                                                                                *
          *                                                                                                             *
          ***************************************************************************************************************
```

```
     UPDATEA_REPLACE (32_REG, A_DESIGNATOR)

REF
PAGE ********************************************************************************************************************
     *                                                                                                                  *
745  *  1    .. UPDATE THE ACCUMULATOR A_DESIGNATOR AND PERFORM REPLACE INSTRUCTION                                      *
746  *  2    ..CHECKS.                                                                                                  *
747  *  3    .. (1) 32_REG - 32 BIT REGISTER CONTAINING VALUE TO BE STORED                                              *
748  *  4    .. (2) A_DESIGNATOR - INTEGER NUMBER SPECIFYING THE ACCUMULATOR (A)                                        *
749  *  5    ..REGISTER TO BE UPDATED                                                                                   *
750   36 *  6    CALL PUT_AREG (A_DESIGNATOR, 32_REG)                                                                  *
751   26 *  7    CALL REPLACE_CHECK (32_REG)                                                                           *
752  *  8    RETURN                                                                                                     *
     *                                                                                                                  *
     ********************************************************************************************************************
```

SET_CO1 (QUANTITY1, QUANTITY2)

```
REF
PAGE ********************************************************************************************************************
     *                                                                                                                    *
     *  1    .. COMPARE QUANTITY1 TO QUANTITY2 SETTING EQUAL/UNEQUAL AND GREATER OR EQUAL/LESS THAN                       *
     *  2    ..(ASR(2,2)) ACCORDINGLY.                                                                                    *
     *  3    .. (1) QUANTITY1 - 32-BIT VALUE.                                                                             *
     *  4    .. (2) QUANTITY2 - 32-BIT VALUE.                                                                             *
     *  5    IF QUANTITY1 = QUANTITY2, THEN                                                                               *
     *  6        ASR(2,1) := 1 ..SET EQUAL INDICATION.                                                                    *
     *  7    ELSE                                                                                                         *
     *  8        ASR(2,1) := 0 ..SET NOT EQUAL INDICATION.                                                                *
     *  9    ENDIF                                                                                                        *
     * 10    IF QUANTITY1 >= QUANTITY2, THEN                                                                              *
     * 11        ASR(1,1) := 1 ..SET GREATER OR EQUAL INDICATION.                                                         *
     * 12    ELSE                                                                                                         *
     * 13        ASR(1,1) := 0 ..SET LESS THAN INDICATION.                                                                *
     * 14    ENDIF                                                                                                        *
     * 15    RETURN                                                                                                       *
     *                                                                                                                    *
     ********************************************************************************************************************
```

```
      SET_CO2 (QUANTITY1, QUANTITY2, QUANTITY3)

      REF
      PAGE ********************************************************************************************************************
           *                                                                                                                    *
771        *  1    .. COMPARE QUANTITY1 TO QUANTITY2 AND QUANTITY3 SETTING OUTSIDE/WITHIN LIMITS                               *
773        *  2    ..(ASR(0,1)) ACCORDINGLY.                                                                                   *
774        *  3    .. (1) QUANTITY1 - 32-BIT REGISTER.                                                                         *
775        *  4    .. (2) QUANTITY2 - 32-BIT REGISTER.                                                                         *
776        *  5    .. (3) QUANTITY3 - 32-BIT REGISTER.                                                                         *
777        *  6    IF QUANTITY3 > QUANTITY1 .AND. QUANTITY1 >= QUANTITY2, THEN                                                 *
778        *  7        ASR(0,1) := 0 ..SET WITHIN LIMITS INDICATION.                                                           *
779        *  8    ELSE                                                                                                        *
780        *  9        ASR(0,1) := 1 ..SET OUTSIDE LIMITS INDICATION.                                                          *
781        * 10    ENDIF                                                                                                       *
782        * 11    RETURN                                                                                                      *
           *                                                                                                                    *
           ********************************************************************************************************************
```

CP UTILITY ROUTINES

```
      ADD_S (S_DESIGNATOR, REF Y)

      REF
      PAGE ********************************************************************************************************************
           *                                                                                                                  *
785        *  1    .. ADD THE APPROPRIATE BASE(S) REGISTER TO THE DISPLACEMENT VALUE(Y).                                      *
786        *  2    .. (1) S_DESIGNATOR - INTEGER NUMBER SPECIFYING THE BASE(S) REGISTER                                       *
787        *  3    ..TO SUPPLY INFORMATION (USED MODULO 8).                                                                   *
788        *  4    .. (2) Y - AN 16-BIT QUANTITY CONTAINING A ZERO EXTENDED 16-BIT VALUE (E.G. U(Y)+8(U(8)))                  *
790        *  5    ..UPON ENTRY. THE 18-BIT RESULT OF THE ADDITION WILL BE RETURNED HERE.                                     *
791        *  6    ..NOTE: THE BASE REGISTER IS NOT ADDED ON NDRO REFERENCES.                                                 *
792        *  7    IF S_DESIGNATOR <> 7 .OR. ASR(7,1)=0 "NOT NDRO REFERENCE", THEN                                            *
793     35 *  8        CALL GET_SREG (MODULO(S_DESIGNATOR,8), 32_REG(1)) ..GET DESIRED BASE REGISTER.                         *
795        *  9        Y := Y + 32_REG(1) ..FORM 18-BIT VALUE.                                                                *
796        * 10    ENDIF                                                                                                      *
797        * 11    RETURN                                                                                                     *
           *                                                                                                                  *
           ********************************************************************************************************************
```

```
      GET_AREG (A_DESIGNATOR, REF 32_REG)

      REF
      PAGE ********************************************************************************************************************
           *                                                                                                                    *
799        *  1    .. RETURN THE CONTENTS OF ACCUMULATOR A_DESIGNATOR IN 32_REG.                                               *
800        *  2    .. (1) A_DESIGNATOR - INTEGER NUMBER SPECIFYING THE ACCUMULATOR (A)                                         *
801        *  3    ..REGISTER TO SUPPLY INFORMATION (USED MODULO 8)                                                            *
802        *  4    .. (2) 32_REG - 32 BIT REGISTER (SUPPLIED BY CALLER) RECEIVING                                              *
803        *  5    ..ACCUMULATOR VALUE                                                                                         *
804        *  6    A_DESIGNATOR := MODULO(A_DESIGNATOR, 8)                                                                     *
805        *  7    IF ASP(10,1) "A/B REGISTER SELECT", THEN                                                                    *
806        *  8        32_REG := CHR(100+A_DESIGNATOR) ..INTERRUPT ACCUMULATOR SELECTED.                                       *
807        *  9    ELSE                                                                                                        *
808        * 10        32_REG := CHR(0+A_DESIGNATOR) ..TASK ACCUMULATOR SELECTED.                                             *
809        * 11    ENDIF                                                                                                       *
810        * 12    RETURN                                                                                                      *
           *                                                                                                                    *
           **********************************************************************************************************************
```

```
      GET_BREG (B_DESIGNATOR, REF 32_REG)

      REF
      PAGE ********************************************************************************************************************
           *                                                                                                                    *
812        *  1    .. LOAD THE LOW ORDER 19-BITS OF 32_REG WITH THE CONTENTS OF THE APPROPRIATE INDEX (B) REGISTER.           *
814        *  2    .. (1) B_DESIGNATOR - INTEGER NUMBER SPECIFYING THE INDEX (B) REGISTER                                     *
815        *  3    ..TO SUPPLY INFORMATION (USED MODULO 8).                                                                   *
816        *  4    .. (2) 32_REG - 32-BIT REGISTER (SUPPLIED BY CALLER) RECEIVING RIGHT                                       *
817        *  5    ..JUSTIFIED 19-BIT QUANTITY FROM THE SPECIFIED INDEX REGISTER.                                             *
818        *  6    B_DESIGNATOR := MODULO(B_DESIGNATOR, 8)                                                                    *
819        *  7    IF B_DESIGNATOR = 0, THEN                                                                                  *
820        *  8        32_REG(18) := 0 ..B(0) IS A SOURCE OF ZEROS.                                                           *
821        *  9    ELSE                                                                                                       *
822        * 10        IF ASR(10,1) "A/B REGISTER SELECT", THEN                                                              *
823        * 11            32_REG(18) := CMR(110+B_DESIGNATOR) ..I.E. APPROPRIATE INTERRUPT INDEX (B) REGISTER.              *
825        * 12        ELSE                                                                                                  *
826        * 13            32_REG(18) := CMR(10+B_DESIGNATOR) ..I.E. APPROPRIATE TASK INDEX (B) REGISTER.                    *
828        * 14        ENDIF                                                                                                 *
829        * 15    ENDIF                                                                                                     *
830        * 16    RETURN                                                                                                    *
           *                                                                                                                    *
           ********************************************************************************************************************
```

```
     GET_SREG (S_DESIGNATOR, REF 32_REG)

     REF
     PAGE ********************************************************************************************************************
          *                                                                                                                    *
632       *  1    .. RETURN THE CONTENTS OF THE BASE REGISTER S_DESIGNATOR IN 32_REG.                                          *
833       *  2    ..(1) S_DESIGNATOR - BASE REGISTER DESIGNATOR                                                               *
834       *  3    ..(2) 32_REG - 32 BIT REGISTER (SUPPLIED BY CALLER) RECEIVING                                               *
835       *  4    ..BASE REGISTER CONTENTS.                                                                                   *
836       *  5    S_DESIGNATOR := MODULO(S_DESIGNATOR, 8)                                                                     *
837       *  6    IF ASR(1$,1) "BASE(S) REGISTER SELECT", THEN                                                                *
838       *  7        32_REG := CHR(120 + S_DESIGNATOR) ..INTERRUPT BASE REGISTER SELECTED.                                   *
840       *  8    ELSE                                                                                                        *
841       *  9        32_REG := CHR(20 + S_DESIGNATOR) ..TASK BASE REGISTER SELECTED.                                         *
843       * 10    ENDIF                                                                                                       *
844       * 11    RETURN                                                                                                      *
          *                                                                                                                    *
          ********************************************************************************************************************
```

```
PUT_AREG (A_DESIGNATOR, 32_REG)

REF
PAGE ********************************************************************************************************************
     *                                                                                                                  *
846  *  1    .. STORE THE CONTENTS OF 32_REG INTO THE ACCUMULATOR A_DESIGNATOR.                                          *
847  *  2    .. (1) A_DESIGNATOR - INTEGER NUMBER SPECIFYING THE ACCUMULATOR (A)                                         *
848  *  3    ..REGISTER TO BE UPDATED (USED MODULO 8)                                                                    *
849  *  4    .. (2) 32_REG - 32 BIT VALUE TO BE STORED.                                                                  *
850  *  5    A_DESIGNATOR := MODULO(A_DESIGNATOR, 8)                                                                     *
851  *  6    IF ASR(10,1) "A/B REGISTER SELECT", THEN                                                                    *
852  *  7        CAR(100+A_DESIGNATOR) := 32_REG ..INTERRUPT ACCUMULATOR SELECTED.                                       *
853  *  8    ELSE                                                                                                        *
854  *  9        CAR(0+A_DESIGNATOR) := 32_REG .. TASK ACCUMULATOR SELECTED.                                             *
855  * 10    ENDIF                                                                                                       *
856  * 11    RETURN                                                                                                      *
     *                                                                                                                  *
     ********************************************************************************************************************
```

```
      PUT_BREG (B_DESIGNATOR, 32_REG)

      REF
      PAGE ********************************************************************************************************************
           *                                                                                                                    *
858        *  1    .. STORE THE LOWER 19 BITS OF 32_REG INTO THE APPROPRIATE INDEX REGISTER.                                    *
860        *  2    .. (1) B_DESIGNATOR - INTEGER NUMBER SPECIFYING THE INDEX (B) REGISTER                                       *
861        *  3    ..TO BE UPDATED (USED MODULO 8).                                                                             *
862        *  4    .. (2) 32_REG - 32-BIT REGISTER CONTAINING RIGHT JUSTIFIED 19-BIT                                            *
863        *  5    ..QUANTITY TO BE STORED INTO THE SPECIFIED INDEX REGISTER.                                                    *
864        *  6    B_DESIGNATOR := MODULO(B_DESIGNATOR, 8)                                                                      *
865        *  7    IF B_DESIGNATOR <> 0, THEN ..ATTEMPT TO STORE INTO B(0) IS A NOOP.                                           *
866        *  8        IF ASR(10,1) "A/B REGISTER SELECT", THEN                                                                 *
867        *  9            CMR(110+B_DESIGNATOR) := 32_REG(18) ..DEFINE INTERRUPT INDEX (B) REGISTER.                           *
869        * 10        ELSE                                                                                                     *
870        * 11            CMR(10+B_DESIGNATOR) := 32_REG(18) ..DEFINE TASK INDEX (B) REGISTER.                                 *
871        * 12        ENDIF                                                                                                    *
872        * 13    ENDIF                                                                                                        *
873        * 14    RETURN                                                                                                       *
           *                                                                                                                    *
           ********************************************************************************************************************
```

```
      PUT_SREG (S_DESIGNATOR, 32_REG)

      DEF
      PAGE ********************************************************************************************************************
           *                                                                                                                    *
875        *  1    .. STORE THE CONTENTS OF 32_REG INTO THE BASE REGISTER S_DESIGNATOR.                                        *
876        *  2    ..(1) S_DESIGNATOR _ BASE REGISTER DESIGNATOR                                                              *
877        *  3    ..(2) 32_REG _ 32 BIT REGISTER CONTAINING VALUE TO BE STORED                                               *
878        *  4    S_DESIGNATOR := MODULO(S_DESIGNATOR, 8)                                                                    *
879        *  5    IF ASR(11,1) "BASE(S) REGISTER SELECT", THEN                                                               *
880        *  6        CMR(120 + S_DESIGNATOR) := 32_REG ..INTERRUPT BASE REGISTER SELECTED.                                  *
882        *  7    ELSE                                                                                                       *
883        *  8        CMR(20 + S_DESIGNATOR) := 32_REG ..TASK BASE REGISTER SELECTED.                                        *
885        *  9    ENDIF                                                                                                      *
886        * 10    RETURN                                                                                                     *
           *                                                                                                                    *
           ********************************************************************************************************************
```

FLOATING POINT SUBROUTINES

```
      FLOATING_ADD_SUBTRACT_HEADER (REF M_MANTISSA, REF A_MANTISSA)

REF
PAGE ********************************************************************************************************************
     *                                                                                                                  *
889  *  1    ..(1) M_MANTISSA - 64 BIT REGISTER SUPPLIED BY CALLER TO CONTAIN MANTISSA FROM MEMORY.                     *
891  *  2    ..(2) A_MANTISSA _ 64-BIT REGISTER SUPPLIED BY CALLER TO CONTAIN MANTISSA FROM A(U(A)+1).                  *
893  *  3    ..                                                                                                        *
894  *  4    SET SPR_PRIVILEGED_INSTRUCTION ..INDICATE PRIVILEGED IF SPR(16,1) SET AND U(I) SET.                       *
896  14 *  5    CALL OP_READ (32_REG(1), "DISPLACE"0) ..GET CHARACTERISTIC FROM MEMORY.                                *
898  14 *  6    CALL OP_READ (M_MANTISSA(63,32), "DISPLACE"1) ..GET MANTISSA FROM MEMORY.                              *
900  *  7    EXTEND SIGN BIT INTO M_MANTISSA(31,32)                                                                    *
901  33 *  8    CALL GET_AREG (U(A), 32_REG(2)) ..GET CHARACTERISTIC FROM A(U(A)).                                        *
902  33 *  9    CALL GET_AREG (U(A)+1, A_MANTISSA(63,32)) ..GET MANTISSA FROM A(U(A)+1).                                  *
904  * 10    EXTEND SIGN BIT INTO A_MANTISSA(31,32)                                                                    *
905  * 11    IF M_MANTISSA=ZERO "NEGATIVE OR POSITIVE", THEN                                                           *
906  * 12        32_REG(1) := 32_REG(2) ..ASSUME INTERMEDIATE CHARACTERISTIC THAT OF ACCUMULATOR VALUE.                *
908  * 13    ENDIF                                                                                                     *
909  * 14    IF A_MANTISSA=ZERO "NEGATIVE OR POSITIVE", THEN                                                           *
910  * 15        32_REG(2) := 32_REG(1) ..ASSUME INTERMEDIATE CHARACTERISTIC THAT OF MEMORY VALUE.                     *
912  * 16    ENDIF                                                                                                     *
913  * 17    SHIFT_COUNT := 32_REG(2)-32_REG(1) ..ONES COMPLEMENT SUBTRACT.                                            *
914  * 18    IF SHIFT_COUNT > 0, THEN                                                                                  *
915  * 19        SHIFT M_MANTISSA RIGHT ARITHMETIC SHIFT_COUNT PLACES                                                  *
916  * 20    ELSE                                                                                                      *
917  * 21        IF SHIFT_COUNT <> 0 "POSITIVE OR NEGATIVE", THEN                                                      *
918  * 22            SHIFT_COUNT := .NOT. SHIFT_COUNT ..ONES COMPLEMENT THE COUNT.                                     *
919  * 23            SHIFT A_MANTISSA RIGHT ARITHMETIC BY SHIFT_COUNT PLACES                                           *
920  * 24        ENDIF                                                                                                 *
921  36 * 25        CALL PUT_AREG (U(A), 32_REG(1)) ..UPDATE INTERMEDIATE CHARACTERISTIC IN A(U(A)).                  *
923  * 26    ENDIF                                                                                                     *
924  * 27    RETURN ..INTERMEDIATE CHARACTERISTIC LEFT IN A(U(A)) FOR LATER USE!                                       *
     *                                                                                                                  *
     ********************************************************************************************************************
```

```
      FLOATING_OVERFLOW (REF 64_REG, REF 32_REG);

      REF
      PAGE ****************************************************************************************************
           *
926        *  1    ..(1) 64_REG - 64-BIT REGISTER SUPPLIED BY CALLER CONTAINING THE INTERMEDIATE RESULT OF A      *
928        *  2    ..FLOATING POINT OPERATION AFTER OVERFLOW HAS OCCURRED.                                         *
929        *  3    ..(2) 32_REG - 32-BIT REGISTER SUPPLIED BY CALLER CONTAINING THE INTERMEDIATE CHARACTERISTIC.   *
931        *  4    SHIFT 64_REG RIGHT 1 BIT ..DIVIDE BY 2.                                                        *
932        *  5    64_REG(63,1) := .NOT.64_REG(63,1) ..RESTORE SIGN BIT.                                          *
933        *  6    32_REG := 32_REG +1 ..INCREMENT THE CHARACTERISTIC.                                            *
934        *  7    RETURN                                                                                         *
           *                                                                                                     *
           ***************************************************************************************************
```

```
      FLOATING_NORMALIZE (REF 64_REG, REF 32_REG)

      REF
      PAGE ********************************************************************************************************************
           *                                                                                                                  *
936        *  1    ..(1) 64_REG - 64-BIT REGISTER SUPPLIED BY CALLER TO BE NORMALIZED AND STORED IN A(U(A)+1).              *
938        *  2    ..(2) 32_REG - 32-BIT REGISTER SUPPLIED BY CALLER CONTAINING INTERMEDIATE CHARACTERISTIC.                *
940        *  3    IF 64_REG(63,1) "NEGATIVE", THEN                                                                         *
941        *  4        SHIFT_COUNT := (# OF LEFT JUSTIFIED 1 BITS IN 64_REG) - 1                                            *
942        *  5    ELSE                                                                                                     *
943        *  6        SHIFT_COUNT := (# LEFT JUSTIFIED 0 BITS IN 64_REG) - 1                                               *
944        *  7    ENDIF                                                                                                    *
945        *  8    IF SHIFT_COUNT < 0°77°, THEN                                                                             *
946        *  9        SHIFT LEFT CIRCULARLY 64_REG BY SHIFT_COUNT                                                          *
947        * 10        32_REG := 32_REG - SHIFT_COUNT ..ADJUST CHARACTERISTIC (ONES COMPLEMENT)                             *
949        * 11    ELSE                                                                                                     *
950        * 12        32_REG := 0                                                                                          *
951        * 13    ENDIF                                                                                                    *
952        * 14    RETURN                                                                                                   *
           *                                                                                                                  *
           ********************************************************************************************************************
```

```
     FLOATING_ROUND (REF 64_REG, REF 32_REG)

     REF
     PAGE ********************************************************************************************************************
          *                                                                                                                    *
954       *  1    ..(1) 64_REG - 64-BIT REGISTER SUPPLIED BY CALLER CONTAINING MANTISSA TO BE ROUNDED.                        *
956       *  2    ..(2) 32_REG - 32-BIT REGISTER SUPPLIED BY CALLER CONTAINING CHARACTERISTIC.                                *
958       *  3    IF 64_REG(31,1)<>64_REG(63,1) "SIGN BIT", THEN                                                              *
959       *  4        IF 64_REG(63,1) "NEGATIVE", THEN                                                                        *
960       *  5            64_REG(63,32) := 64_REG(63,32)-1                                                                    *
961       *  6        ELSE                                                                                                    *
962       *  7            64_REG(63,32) := 64_REG(63,32)+1                                                                    *
963       *  8        ENDIF                                                                                                   *
964       *  9        IF OVERFLOW OCCURRED, THEN                                                                              *
965    41 * 10            CALL FLOATING_OVERFLOW (64_REG, 32_REG)                                                             *
966       * 11        ENDIF                                                                                                   *
967       * 12    ENDIF                                                                                                       *
968       * 13    RETURN                                                                                                      *
          *                                                                                                                    *
          ********************************************************************************************************************
```

NSWC AN/UYK-7 (CP)
FLOATING POINT SUBROUTINES

```
      ROUND_UP (REF MANTISSA, REF CHARACTERISTIC)

      REF
      PAGE ********************************************************************************************************************
           *                                                                                                                    *
           *  1    ..(1) MANTISSA - 32-BIT REGISTER SUPPLIED BY CALLER CONTAINING MANTISSA TO BE ROUNDED UP.                   *
970        *  2    ..(2) CHARACTERISTIC - 32-BIT REGISTER SUPPLIED BY CALLER CONTAINING CHARACTERISTIC.                        *
972        *  3    MANTISSA := MANTISSA+1                                                                                     *
974        *  4    IF MANTISSA(31,1)=1 "OVERFLOW OCCURRED", THEN                                                              *
975        *  5        ..NOTE THAT WORKING ONLY WITH POSITIVE QUANTITIES.                                                     *
976        *  6        MANTISSA := MANTISSA.RL.1 ..ADJUST THE MANTISSA.                                                       *
977        *  7        CHARACTERISTIC := CHARACTERISTIC+1 ..UPDATE THE CHARACTERISTIC.                                        *
978        *  8    ENDIF                                                                                                      *
979        *  9    RETURN                                                                                                     *
980        *                                                                                                                  *
           ********************************************************************************************************************
```

```
      DIVIDE_COMPARE (REF 64_REG, 32_REG)

      REF
      PAGE ********************************************************************************************************************
           *                                                                                                                    *
982        *  1    ..(1) 64_REG - 64-BIT REGISTER SUPPLIED BY CALLER CONTAINING DIVIDEND.                                      *
983        *  2    ..(2) 32_REG - 32-BIT REGISTER CONTAINING DIVISOR.                                                          *
984        *  3    IF 32_REG<=64_REG(63,32), THEN ..USE 32-BIT UNSIGNED COMPARE.                                               *
985        *  4        64_REG(63,32) := 64_REG(63,32) - 32_REG ..ONES COMPLEMENT SUBTRACT.                                     *
986        *  5        64_REG(0,1) := 1                                                                                        *
987        *  6    ENDIF                                                                                                       *
988        *  7    RETURN                                                                                                      *
           *                                                                                                                    *
           ********************************************************************************************************************
```

```
     FLOATING_POINT_END (MANTISSA, CHARACTERISTIC)

     REF
     PAGE ********************************************************************************************************************
          *                                                                                                                    *
990       *  1     .. COMMON CODE FOR TERMINATION OF FLOATING POINT INSTRUCTIONS.                                              *
991       *  2     ..(1) MANTISSA - 32-BIT RESULTANT MANTISSA                                                                 *
992       *  3     ..(2) CHARACTERISTIC - 32-BIT RESULTANT CHARACTERISTIC.                                                    *
993       *  4     IF MANTISSA=0 "RESULT ZERO", THEN                                                                          *
994       *  5         CHARACTERISTIC := 0 ..ZERO THE CHARACTERISTIC.                                                         *
995       *  6     ENDIF                                                                                                      *
996    36 *  7     CALL PUT_AREG (U(A)+1, MANTISSA)                                                                           *
997    36 *  8     CALL PUT_AREG (U(A), CHARACTERISTIC)                                                                       *
998       *  9     IF CHARACTERISTIC(15,1)<>CHARACTERISTIC(31,1), THEN                                                        *
999    11 * 10         CALL GENERATE_SYNCHRONOUS_INTERRUPT ("P-"1, "FLOATING POINT OVERFLOW)                                  *
1000      * 11         ..ABORT THE INSTRUCTION..                                                                              *
1001      * 12     ENDIF                                                                                                      *
1002      * 13     RETURN                                                                                                     *
          *                                                                                                                    *
          **********************************************************************************************************************
```

CP INSTRUCTION SET

_OR(& _ROR) ..(REPLACE) INCLUSIVE OR FMT II F=01 D & F=03 D

```
REF
PAGE ********************************************************************************************************************
     *
1005   14 *  1    CALL OP_READ(32_REG(1),"DISPLACE"0) ..ACQUIRE OPERAND (Y) AND A(U(A)).                                  *
1006   33 *  2    CALL GET_AREG(U(A), REPEAT_ACCUMULATOR)                                                                 *
1007      *  3    REPEAT_ACCUMULATOR := REPEAT_ACCUMULATOR .V. 32_REG(1) ..PERFORM INCLUSIVE OR.                          *
1009   28 *  4    CALL UPDATEA_REPLACE(REPEAT_ACCUMULATOR, U(A)) ..RESULT GOES TO CMR & POSSIBLY TO MEMORY.               *
1011      *  5    RETURN                                                                                                  *
     *                                                                                                                   *
     *                                                                                                                   *
     *********************************************************************************************************************
```

_SC(S _RSC) ..(REPLACE) SELECTIVE CLEAR A FMT II F=01 1 & F=03 1

```
REF
PAGE ********************************************************************************************************************
     *                                                                                                                    *
1013   14 *  1    CALL OP_READ (32_REG(1), "DISPLACE"0) ..FETCH OPERAND (Y) AND A(U(A)).                                  *
1015      *  2    32_REG(1) := .NOT. 32_REG(1) ..COMPLEMENT Y.                                                            *
1016   33 *  3    CALL GET_AREG(U(A), REPEAT_ACCUMULATOR)                                                                 *
1017      *  4    REPEAT_ACCUMULATOR := REPEAT_ACCUMULATOR .A. 32_REG(1)                                                  *
1018   28 *  5    CALL UPDATEA_REPLACE (REPEAT_ACCUMULATOR, U(A)) ..RESULT GOES CMR & POSSIBLY TO MEMORY.                 *
1020      *  6    RETURN                                                                                                  *
     *                                                                                                                    *
     ********************************************************************************************************************
```

_MS(E _PMS) ..(REPLACE) SELECTIVE SUBSTITUTE FMT II F=01 2 E F=03 2

```
REF
PAGE ********************************************************************************************************************
     *                                                                                                                  *
1022   14 *  1    CALL OP_READ(32_REG(1), "DISPLACE"0) ..FETCH OPERAND (Y), A(U(A)), & A(U(A)+1).                       *
1024   33 *  2    CALL GET_AREG(U(A), 32_REG(2))                                                                        *
1025   33 *  3    CALL GET_AREG(U(A)+1, REPEAT_ACCUMULATOR)                                                             *
1026      *  4    32_REG(1) := 32_REG(1) .A. 32_REG(2)                                                                  *
1027      *  5    REPEAT_ACCUMULATOR := (REPEAT_ACCUMULATOR .A. (.NOT.32_REG(2))) .V. 32_REG(1)                         *
1029   28 *  6    CALL UPDATEA_REPLACE(REPEAT_ACCUMULATOR, U(A)+1) ..RESULT GOES TO A(U(A)+1) & POSSIBLY TO MEMORY.     *
1031      *  7    RETURN                                                                                                *
     *                                                                                                                  *
     ********************************************************************************************************************
```

_XOR(E _RXOP) ..(REPLACE) EXCLUSIVE OR FMT II F=01 3 & F=03 3

```
REF
PAGE ********************************************************************************************************************
     *                                                                                                                  *
1033   14 *  1    CALL GP_READ(32_REG(1),"DISPLACE"0) ..FETCH OPERAND (Y) & A(U(A)).                                     *
1035   33 *  2    CALL GET_AREG(U(A), REPEAT_ACCUMULATOR)                                                                *
1036      *  3    REPEAT_ACCUMULATOR := REPEAT_ACCUMULATOR .X. 32_REG(1) ..PERFORM EXCLUSIVE OR.                         *
1038   28 *  4    CALL UPDATEA_REPLACE(REPEAT_ACCUMULATOR, U(A)) ..RESULT GOES TO A(U(A)) & POSSIBLY TO MEMORY.          *
1040      *  5    RETURN                                                                                                 *
     *                                                                                                                  *
     ********************************************************************************************************************
```

```
     _ALP(6 _RALP) ..(REPLACE) ADD LOG. PROD. FMT II F=01 4 & F=03 4

     REF
     PAGE ********************************************************************************************************************
          *                                                                                                                    *
1042   14 *  1    CALL OP_READ(REPEAT_ACCUMULATOR,"DISPLACE"0) ..FETCH OPERAND (Y), A(U(A)) & A(U(A)+1).                       *
1044   33 *  2    CALL GET_AREG(U(A), 32_REG(2))                                                                               *
1045   33 *  3    CALL GET_AREG(U(A)+1, 32_REG(3))                                                                             *
1046      *  4    REPEAT_ACCUMULATOR := (REPEAT_ACCUMULATOR .A. 32_REG(2)) + 32_REG(3)                                         *
1047      *  5    UPDATE FIXED POINT OVERFLOW(ASR(3,1))                                                                        *
1048   28 *  6    CALL UPDATEA_REPLACE(REPEAT_ACCUMULATOR, U(A)+1) ..RESULT GOES TO CAR & POSSIBLY TO MEMORY.                  *
1050      *  7    RETURN                                                                                                       *
          *                                                                                                                    *
          ********************************************************************************************************************
```

```
     _LLP (_NLP, _LLPN, _RLP, & _RNLP)

     REF
     PAGE ********************************************************************************************************************
          *                                                                                                                    *
1052      *  1    .. LOAD, SUBTRACT, AND REPLACE LOGICAL PRODUCT.                                                            *
1053      *  2    ..FMT II, F=01 5, 01 6, 01 7, 03 5, 03 6                                                                   *
1054   14 *  3    CALL OP_READ (REPEAT_ACCUMULATOR, "DISPLACE"0)                                                             *
1055   33 *  4    CALL GET_AREG (U(A), 32_REG(3));                                                                           *
1056      *  5    REPEAT_ACCUMULATOR := 32_REG(3) .A. REPEAT_ACCUMULATOR ..FORM LOGICAL PRODUCT.                             *
1058      *  6    IF _NLP .OR. _RNLP, THEN                                                                                   *
1059   33 *  7        CALL GET_AREG (U(A)+1, 32_REG(2))                                                                      *
1060      *  8        REPEAT_ACCUMULATOR := 32_REG(2) - REPEAT_ACCUMULATOR ..ONES COMPLEMENT SUBTRACT.                       *
1062      *  9        UPDATE FIXED POINT OVERFLOW BIT IN ASR(BIT 3)                                                          *
1063      * 10    ENDIF                                                                                                      *
1064      * 11    IF _LLP, THEN                                                                                              *
1065   36 * 12        CALL PUT_AREG (U(A), REPEAT_ACCUMULATOR)                                                               *
1066      * 13    ELSE                                                                                                       *
1067   28 * 14        CALL UPDATEA_REPLACE(REPEAT_ACCUMULATOR, U(A)+1)                                                       *
1068      * 15    ENDIF                                                                                                      *
1069      * 16    RETURN                                                                                                     *
          *                                                                                                                    *
          ********************************************************************************************************************
```

```
     _CNT ..COUNT ONES, FMT II, F=02 0

REF
PAGE ********************************************************************************************************************************
     *
1071   14 *  1    CALL OP_READ (32_REG(2), "DISPLACE=0)                                                                            *
1072      *  2    REPEAT_ACCUMULATOR := 0                                                                                          *
1073      *  3    I := 0                                                                                                           *
1074      *  4    DO _WHILE I < 32 ..LOOP AND COUNT ONES.                                                                          *
1075      *  5        IF 32_REG(2)(I,1), THEN                                                                                      *
1076      *  6            REPEAT_ACCUMULATOR := REPEAT_ACCUMULATOR + 1                                                             *
1077      *  7        ENDIF                                                                                                        *
1078      *  8        I := I + 1                                                                                                   *
1079      *  9    ENDDO                                                                                                            *
1080   36 * 10    CALL PUT_AREG (U(A), REPEAT_ACCUMULATOR) ..STORE RESULTS                                                         *
1081      * 11    RETURN                                                                                                           *
     *                                                                                                                            *
     ********************************************************************************************************************************
```

```
     _XR (_XRL) ..EXECUTE REMOTE (LOWER), FMT II, F=02 2, F=02 3

REF
PAGE ***************************************************************************************************************
     *                                                                                                             *
1083 *  1    .. _XR EXECUTES ONE INSTRUCTION ONLY (I. E. ONE FULL WORD INSTRUCTION                                   *
1084 *  2    ..OR UPPER HALF OF TWO HALF WORD INSTRUCTIONS).                                                        *
1085   13 *  3    CALL JUMP_ADDRESS("DISPLACE"0, OPERAND_S, OPERAND_DISPLACEMENT)                                   *
1086 *  4    ..DO OPERAND CHECKS FOR JUMP TYPE INSTRUCTIONS. OPERAND BASE REGISTER                                  *
1087 *  5    ..AND OPERAND DISPLACEMENT ARE RETURNED.                                                               *
1088 *  6    32_REG(3) := OPERAND_DISPLACEMENT                                                                      *
1089   32 *  7    CALL ADD_S (OPERAND_S, 32_REG(3)) ..FIND REMOTE INSTRUCTIONS ABSOLUTE ADDRESS.                     *
1091   17 *  8    CALL BPR_CHECK (32_REG(3), INSTRUCTION) .. PERFORM INSTRUCTION BREAKPOINT CHECKS.                  *
1093   20 *  9    CALL SPR_CHECK (U(S), INSTRUCTION EXECUTE, "P-"1, OPERAND_DISPLACEMENT)                            *
1094 * 10    ..PERFORM INSTRUCTION SPR CHECKS                                                                       *
1095 * 11    OLDF2 := U(F2) ..SAVE _XR, _XRL INDICATOR.                                                             *
1096   19 * 12    CALL MEMORY_READ (32_REG(3), U, U(S), INSTRUCTION, "P-"1)                                         *
1097 * 13    ..FETCH REMOTE INSTRUCTION INTO U REGISTER.                                                            *
1098 * 14    IF OLDF2 ODD, THEN .. ODD => _XRL.                                                                     *
1099 * 15        UU := UL ..COPY LOWER HALF WORD INSTRUCTION TO UU.                                                 *
1100 * 16    ENDIF                                                                                                  *
1101 * 17    SET EXECUTE_REMOTE_IN_PROGRESS                                                                         *
1102 * 18    DEFINE INSTRUCTION_FORMAT_INDICATOR ..I.E. I, II, III, OR IV.                                          *
1103 * 19    CALL _DECODE ..TRANSFER CONTROL TO APPROPRIATE INSTRUCTION.                                            *
1104 * 20    RETURN                                                                                                 *
     *                                                                                                             *
     ***************************************************************************************************************
```

```
_SLP ..STORE LOGICAL PRODUCT, FMT II, F=02 4

REF
PAGE ********************************************************************************************************************
     *                                                                                                                  *
1106   33 *  1    CALL GET_AREG (U(A), REPEAT_ACCUMULATOR)                                                             *
1107   33 *  2    CALL GET_AREG (U(A)+1, 32_REG(2))                                                                    *
1108      *  3    REPEAT_ACCUMULATOR := REPEAT_ACCUMULATOR .A. 32_REG(2)                                               *
1109   15 *  4    CALL OP_STORE (REPEAT_ACCUMULATOR, "DISPLACE"0)                                                      *
1110      *  5    RETURN                                                                                               *
     *                                                                                                                  *
     ********************************************************************************************************************
```

```
_SSUM ..STORE SUM, FMT II, F=02 5

REF
PAGE ********************************************************************************************************************************
     *
1112   33 *  1    CALL GET_AREG (U(A), REPEAT_ACCUMULATOR)                                                                          *
1113   33 *  2    CALL GET_AREG (U(A)+1, 32_REG(2))                                                                                 *
1114      *  3    REPEAT_ACCUMULATOR := REPEAT_ACCUMULATOR + 32_REG(2) ..ONES COMPLEMENT ADDITION.                                  *
1116      *  4    IF OVERFLOW OCCURRED, THEN                                                                                        *
1117      *  5        ASR(3,2) := 1 ..SET FIXED POINT OVERFLOW INDICATOR.                                                           *
1118      *  6    ELSE                                                                                                              *
1119      *  7        ASR(3,1) := 0 ..CLEAR FIXED POINT OVERFLOW INDICATOR.                                                         *
1120      *  8    ENDIF                                                                                                             *
1121   36 *  9    CALL PUT_AREG ( U(A)+1, REPEAT_ACCUMULATOR)                                                                       *
1122   15 * 10    CALL CP_STORE (REPEAT_ACCUMULATOR, "DISPLACE"0)                                                                   *
1123      * 11    RETURN                                                                                                            *
     *                                                                                                                             *
     ********************************************************************************************************************************
```

```
_SDIF .. STORE DIFFERENCE, FMT II, F=02 6

REF
PAGE ********************************************************************************************************************************
     *
1125   33 *  1    CALL GET_AREG(U(A),32_REG(1))                                                                                      *
1126   33 *  2    CALL GET_AREG(U(A)+1,REPEAT_ACCUMULATOR)                                                                           *
1127      *  3    REPEAT_ACCUMULATOR := REPEAT_ACCUMULATOR - 32_REG(1) ..ONES COMPLEMENT.                                            *
1128      *  4    IF OVERFLOW OCCURRED, THEN                                                                                         *
1129      *  5        ASR(3,1) := 1 ..SET FIXED POINT OVERFLOW INDICATOR.                                                            *
1130      *  6    ELSE                                                                                                               *
1131      *  7        ASR(3,1) := 0 ..CLEAR FIXED POINT OVERFLOW INDICATOR.                                                          *
1132      *  8    ENDIF                                                                                                              *
1133   36 *  9    CALL PUT_AREG(U(A)+1,REPEAT_ACCUMULATOR)                                                                           *
1134   15 * 10    CALL OP_STORE(REPEAT_ACCUMULATOR,"DISPLACE"0)                                                                      *
1135      * 11    RETURN                                                                                                             *
     *                                                                                                                              *
     ********************************************************************************************************************************
```

_DS .. DOUBLE STORE ACCUMULATORS, FMT II, F=02 7

```
REF
PAGE ********************************************************************************************************************
     *                                                                                                                    *
1137 *  1    SET SPR_PRIVILEGED_INSTRUCTION ..INDICATE PRIVILEGED IF SPR(16,1) SET AND U(I) SET.                          *
1139   33 *  2    CALL GET_AREG(U(A),32_REG(1))                                                                          *
1140   33 *  3    CALL GET_AREG(U(A)+1,32_REG(2))                                                                        *
1141   15 *  4    CALL OP_STORE(32_REG(1),"DISPLACE"0)                                                                   *
1142   15 *  5    CALL OP_STORE(32_REG(2),"DISPLACE"1)                                                                   *
1143 *  6    RETURN                                                                                                      *
     *                                                                                                                    *
     ********************************************************************************************************************
```

```
_TSF .. TEST & SET FLAG, FMT II, F=03 7

REF
PAGE ********************************************************************************************************************
     *                                                                                                                  *
1145 *  1    IF U(16,1) "INDIRECT ADDRESSING", THEN ..NOT ALLOWED (AS PER REP CARD1).                                   *
1147 * 11 *  2       CALL GENERATE_SYNCHRONOUS_INTERRUPT("P-"1, CP ILLEGAL INST) ..ABORT THE INSTRUCTION.               *
1149 *  3    ENDIF                                                                                                      *
1150 * 14 *  4    CALL OP_READ(REPEAT_ACCUMULATOR, "DISPLACE" 0)                                                        *
1151 *  5    IF REPEAT_ACCUMULATOR(31,1), THEN                                                                          *
1152 *  6       ASR(2,1) := 0 ..NOT EQUAL.                                                                             *
1153 *  7    ELSE                                                                                                       *
1154 *  8       ASR(2,1) := 1 ..EQUAL.                                                                                 *
1155 *  9    ENDIF                                                                                                      *
1156 * 10    REPEAT_ACCUMULATOR(31,1) := 1 ..SET FLAG BIT.                                                              *
1157 * 15 * 11    CALL OP_STORE(REPEAT_ACCUMULATOR, "DISPLACE" 0)                                                       *
1158 * 12    RETURN                                                                                                     *
     *                                                                                                                  *
     ********************************************************************************************************************
```

```
     _DL .. DOUBLE LOAD, FMT II F=05 0

     REF
     PAGE ********************************************************************************************************************
          *
1160      *  1     SET SPR_PRIVILEGED_INSTRUCTION ..INDICATE PRIVILEGED IF SPR(16,1) SET AND U(I) SET.                        *
1161   14 *  2     CALL OP_READ(32_REG(1),'DISPLACE=0)                                                                        *
1163   14 *  3     CALL OP_READ(32_REG(2),'DISPLACE=1)                                                                        *
1164   36 *  4     PUT_AREG(U(A),32_REG(1))                                                                                   *
1165   36 *  5     PUT_AREG(U(A)+1,32_REG(2))                                                                                 *
1166      *  6     RETURN                                                                                                     *
          *                                                                                                                  *
          ********************************************************************************************************************
```

DA (& DAN) .. DOUBLE ADD & SUBTRACT, FMT II, F=05 1 & F=05 2

```
REF
PAGE ********************************************************************************************************************
     *
1168 *  1    .. NOTE: 64_REG(1) AND 64_REG(2) ARE 64-BIT REGISTERS USED FOR DOUBLE LENGTH INSTRUCTION.
1170 *  2    SET SPR_PRIVILEGED_INSTRUCTION ..INDICATE PRIVILEGED IF SPR(16,1) SET AND U(I) SET.
1172   14 *  3    CALL OP_READ(64_REG(2)(31,32),"DISPLACE=0)
1173   14 *  4    CALL OP_READ(64_REG(2)(63,32),"DISPLACE=1)
1174   33 *  5    GET_AREG(U(A),64_REG(1)(31,32))
1175   33 *  6    GET_AREG(U(A)+1,64_REG(1)(63,32))
1176 *  7    IF U(F2) = 1, THEN ..ADD.
1177 *  8        64_REG(1) := 64_REG(1) + 64_REG(2) ..ONES COMPLEMENT DOUBLE ADD.
1178 *  9    ELSE ..SUBTRACT.
1179 * 10        64_REG(1) := 64_REG(1) - 64_REG(2) ..ONES COMPLEMENT DOUBLE SUBTRACT.
1180 * 11    ENDIF
1181 * 12    IF OVERFLOW, THEN
1182 * 13        ASR(3,1) := 1 ..SET FIXED POINT OVERFLOW INDICATOR.
1183 * 14    ELSE
1184 * 15        ASR(3,1) := 0 ..CLEAR FIXED POINT OVERFLOW INDICATOR.
1185 * 16    ENDIF
1186   36 * 17    PUT_AREG(U(A),64_REG(1)(31,32))
1187   36 * 18    PUT_AREG(U(A)+1,64_REG(1)(63,32))
1188 * 19    RETURN
     *
     ********************************************************************************************************************
```

```
     _DC .. DOUBLE COMPARE, FMT II, F= 05 3

     REF
     PAGE ********************************************************************************************************************
          *
1190      *  1     .. NOTE: 64_REG(1) AND 64_REG(2) ARE 64-BIT REGISTERS USED FOR DOUBLE                                    *
1191      *  2     ..LENGTH INSTRUCTIONS.                                                                                   *
1192      *  3     SET SPR_PRIVILEGED_INSTRUCTION ..INDICATE PRIVILEGED IF SPR(16,1) SET AND U(I) SET.                      *
1194    33 *  4     CALL GET_AREG(U(A)+1,64_REG(1)(63,32))                                                                   *
1195    33 *  5     CALL GET_AREG(U(A),64_REG(1)(31,32))                                                                     *
1196    14 *  6     CALL OP_READ(64_REG(2)(63,32),"DISPLACE=1)                                                               *
1197    14 *  7     CALL OP_READ(64_REG(2)(31,32),"DISPLACE=0)                                                               *
1198      *  8     IF 64_REG(1) = 64_REG(2), THEN                                                                           *
1199      *  9         ASR(2,1) := 1 ..SET EQUAL INDICATOR.                                                                 *
1200      * 10     ELSE                                                                                                     *
1201      * 11         ASR(2,1) := 0 ..INDICATE NOT EQUAL.                                                                  *
1202      * 12     ENDIF                                                                                                    *
1203      * 13     IF 64_REG(1) >= 64_REG(2), THEN                                                                          *
1204      * 14         ASR(1,1) := 1 ..INDICATE GREATER THAN OR EQUAL.                                                      *
1205      * 15     ELSE                                                                                                     *
1206      * 16         ASR(1,1) := 0 ..INDICATE LESS THAN.                                                                  *
1207      * 17     ENDIF                                                                                                    *
1208      * 18     RETURN                                                                                                   *
          *                                                                                                               *
          *****************************************************************************************************************
```

```
     _LBMP ..LOAD BASE & MEMORY PROTECTION, FMT II F=05 4

     REF
     PAGE ********************************************************************************************************************
          *
1210   14 *  1    CALL OP_READ (32_REG(1), "DISPLACE"0) ..INDIRECTION, SPR &BPR CHECKS. U(19,20) IS NOW UPDATED.          *
1212   14 *  2    CALL OP_READ (32_REG(2), "DISPLACE"1)                                                                   *
1213   34 *  3    CALL GET_BREG (U(B), 32_REG(3))                                                                         *
1214      *  4    32_REG(3) := 32_REG(3)(15,16) + U(Y)                                                                    *
1215      *  5    IF 32_REG(3)(0,1) "ODD ADDRESS", THEN                                                                   *
1216   11 *  6        CALL GENERATE_SYNCHRONOUS_INTERRUPT ("P-"1, CP ILLEGAL INSTRUCTION)                                 *
1217      *  7        ..ABORT INSTRUCTION.                                                                                *
1218      *  8    ELSE                                                                                                    *
1219      *  9        IF ASR(19,4)=0 "TASK MODE" .AND. (.NOT. ASR(0,1) "NOT LOAD BASE ENABLE" .OR.                        *
1221      * 10            U(S)<>7 .OR. U(A)=7), THEN                                                                     *
1222   11 * 11            CALL GENERATE_SYNCHRONOUS_INTERRUPT ("P-"1, PRIVILEGED INSTRUCTION VIOLATION)                  *
1224      * 12            ..ABORT INSTRUCTION.                                                                           *
1225      * 13        ENDIF                                                                                              *
1226      * 14        CMR(0'20'+U(A)) := 32_REG(1)(17,18) ..SET UP THE BASE REGISTER.                                    *
1227      * 15        CMR(0'160'+U(A)) := 32_REG(2)(20,21) ..SET UP ASSOCIATED STORAGE PROTECTION REGISTER.              *
1229      * 16        CMR(0'170'+U(A))(19,3) := U(S) ..SET UP BASE DESIGNATOR OF ASSOCIATED SIR.                         *
1231      * 17        CMR(0'170'+U(A))(15,16) := 32_REG(3)(15,16) ..SET UP DISPLACEMENT OF ASSOCIATED SIR.               *
1233      * 18    ENDIF                                                                                                  *
1234      * 19    RETURN                                                                                                 *
          *                                                                                                            *
          ********************************************************************************************************************
```

```
_FA (_FAR) ..FLOATING-POINT ADD (WITH ROUND), FMT II, F=06 0 & 06 4

REF
PAGE ********************************************************************************************************************
     *                                                                                                                  *
1236 *   1    CALL FLOATING_ADD_SUB_HDR (64_REG(1), 64_REG(2))                                                          *
1237 *   2    64_REG(2) := 64_REG(2) + 64_REG(1) ..64-BIT ONE'S COMPLEMENT ADD.                                         *
1238   33 *   3    CALL GET_AREG (U(A), 32_REG(1)) ..GET INTERMEDIATE CHARACTERISTIC.                                    *
1239 *   4    IF OVERFLOW OCCURRED, THEN                                                                                *
1240   41 *   5        CALL FLOATING_OVERFLOW (64_REG(2), 32_REG(1))                                                     *
1241 *   6    ELSE                                                                                                      *
1242   42 *   7        CALL FLOATING_NORMALIZE (64_REG(2), 32_REG(1))                                                    *
1243 *   8    ENDIF                                                                                                     *
1244 *   9    IF U(F2)=4 "ROUNDING OPTION", THEN                                                                        *
1245   43 *  10        CALL FLOATING_ROUND (64_REG(2), 32_REG(1))                                                        *
1246 *  11    ENDIF                                                                                                     *
1247   46 *  12    CALL FLOATING_POINT_END (64_REG(2)(63:32)), 32_REG(1))                                                *
1248 *  13    RETURN                                                                                                    *
     *                                                                                                                  *
     ********************************************************************************************************************
```

```
_FAN (_FANR) ..FLOATING-POINT SUBTRACT (ROUNDD), FMT II, F=C6 1 C 06 5

REF
PAGE ********************************************************************************************************************************
     *                                                                                                                              *
1250 *  1    CALL FLOATING_ADD_SUB_MOR (64_REG(1), 64_REG(2))                                                                       *
1251 *  2    64_REG(2) := 64_REG(2) - 64_REG(1) ..64-BIT ONE'S COMPLIMENT SUBTRACT.                                                 *
1252  33 *  3    CALL GET_AREG (U(A), 32_REG(1)) ..GET INTERMEDIATE CHARACTERISTIC.                                                 *
1253 *  4    IF OVERFLOW OCCURRED, THEN                                                                                             *
1254  41 *  5        CALL FLOATING_OVERFLOW (64_REG(2), 32_REG(1))                                                                  *
1255 *  6    ELSE                                                                                                                   *
1256  42 *  7        CALL FLOATING_NORMALIZE (64_REG(2), 32_REG(1))                                                                 *
1257 *  8    ENDIF                                                                                                                  *
1258 *  9    IF U(F2)=F "ROUNDING OPTION", THEN                                                                                     *
1259  43 * 10        CALL FLOATING_ROUND (64_REG(2), 32_REG(1))                                                                     *
1260 * 11    ENDIF                                                                                                                  *
1261  46 * 12    CALL FLOATING_POINT_END (64_REG(2)(63,32), 32_REG(1))                                                              *
1262 * 13    RETURN                                                                                                                 *
     *                                                                                                                              *
     ********************************************************************************************************************************
```

```
      _FN (_FMR) ..FLOATING POINT MULTIPLY (ROUND), FMT II, F=06 2 & 06 6

      REF
      PAGE ********************************************************************************************************************
           *                                                                                                                    *
1264       *  1    SET SPR_PRIVILEGED_INSTRUCTION ..INDICATE PRIVILEGED IF SPR(16,1) SET AND U(I) SET.                         *
1266    14 *  2    CALL OP_READ (32_REG(4), "DISPLACE"0) ..GET MULTIPLIER CHARACTERISTIC FROM MEMORY.                           *
1268    14 *  3    CALL OP_READ (32_REG(1), "DISPLACE"1) ..GET MULTIPLIER MANTISSA FROM MEMORY.                                 *
1270    33 *  4    CALL GET_AREG (U(A), 32_REG(3)) ..GET MULTIPLICAND CHARACTERISTIC FROM A(U(A)).                             *
1272    33 *  5    CALL GET_AREG (U(A)+1, 32_REG(2)) ..GET MULTIPLICAND MANTISSA FROM A(U(A)+1).                               *
1274       *  6    SIGN_INDICATOR := 32_REG(1)(31,1) .XOR. 32_REG(2)(31,1) ..SAVE SIGN FOR RESULT.                             *
1276       *  7    FORCE 32_REG(1) "MULTIPLIER" POSITIVE                                                                       *
1277       *  8    IF 32_REG(1) = 0 , THEN ..MULTIPLY BY ZERO.                                                                 *
1278    36 *  9        CALL PUT_AREG (U(A), 32_REG(1))                                                                         *
1279    36 * 10        CALL PUT_AREG (U(A)+1, 32_REG(1))                                                                       *
1280       * 11        RETURN                                                                                                  *
1281       * 12    ENDIF                                                                                                       *
1282       * 13    32_REG(3) := 32_REG(3) + 32_REG(4) ..FORM THE NEW CHARACTERISTIC.                                           *
1283    96 * 14    CALL DO_MULTIPLY ..32_REG(2) X 32_REG(1) WITH 64-BIT RESULT PUT IN REPEAT_ACCUMULATOR &                    /*
                                 32_REG(1).                                                                                  *
1285       * 15    SHIFT REPEAT_ACCUMULATOR AND 32_REG(1) PAIR LEFT LOGICAL ONE ..CORRECT FOR FRACTIONAL MULTIPLY.             *
1287       * 16    IF REPEAT_ACCUMULATOR(30,1) = 0 , THEN ..NORMALIZE.                                                         *
1288       * 17        SHIFT REPEAT_ACCUMULATOR AND 32_REG(1) PAIR LEFT LOGICAL ONE                                            *
1289       * 18        32_REG(3) := 32_REG(3) - 1 ..UPDATE CHARACTERISTIC.                                                     *
1290       * 19    ENDIF                                                                                                       *
1291       * 20    IF U(F2) = 6 "ROUNDING OPTION", THEN                                                                        *
1292    43 * 21        CALL FLOATING_ROUND (REPEAT_ACCUMULATOR, 32_REG(3))                                                     *
1293       * 22    ENDIF                                                                                                       *
1294       * 23    IF SIGN_INDICATOR "NEGATIVE", THEN ..DO SIGN CORRECTION.                                                    *
1295       * 24        REPEAT_ACCUMULATOR := .NOT. REPEAT_ACCUMULATOR                                                          *
1296       * 25    ENDIF                                                                                                       *
1297    46 * 26    CALL FLOATING_POINT_END (REPEAT_ACCUMULATOR, 32_REG(3))                                                     *
1298       * 27    RETURN                                                                                                      *
           *                                                                                                                    *
           ********************************************************************************************************************
```

_FDI(_FDR) ..FLOATING-POINT DIVIDE (WITH ROUND), FMT II, F=06 3 & 06 7

```
REF
PAGE ************************************************************************************************************************
     *                                                                                                                      *
1300 *   1    SET SPR_PRIVILEGED_INSTRUCTION ..INDICATE PRIVILEGED IF SPR(16,1) SET AND U(I) SET.                          *
1302 14 *   2    CALL OP_READ (32_REG(2), "DISPLACE"0) ..GET DIVISOR CHARACTERISTIC FROM MEMORY.                          *
1304 14 *   3    CALL OP_READ (32_REG(1), "DISPLACE"1) ..GET DIVISOR MANTISSA FROM MEMORY.                                *
1306 33 *   4    CALL GET_AREG (U(A), 32_REG(3)) ..GET DIVIDEND CHARACTERISTIC FROM A(U(A)).                              *
1308 33 *   5    CALL GET_AREG(U(A)+1, 64_REG(1)(63,32)) .. GET DIVIDEND MANTISSA FROM A(U(A)+1).                         *
1310 *   6    SIGN_IND := 64_REG(1)(63,1) .XOR. 32_REG(1)(31,1) ..SIGN INDICATOR.                                          *
1311 *   7    FORCE 64_REG(1) "DIVIDEND" AND 32_REG(1) "DIVISOR" POSITIVE                                                  *
1312 *   8    IF 64_REG(1)(63,32) "DIVIDEND" = 0, THEN                                                                     *
1313 *   9        32_REG(3) := 0 ..FORCE DIVIDEND CHARACTERISTIC TO ZERO.                                                  *
1314 *  10    ENDIF                                                                                                        *
1315 *  11    IF 32_REG(1) = 0 "DIVISOR ZERO", THEN                                                                        *
1316 11 *  12        CALL GENERATE_SYNCHRONOUS_INTERRUPT ("P-"1, FLOATING POINT OVERFLOW)                                 *
1317 *  13    ENDIF                                                                                                        *
1318 *  14    32_REG(3) := 32_REG(3) - 32_REG(2) ..COMPUTE INTERMEDIATE CHARACTERISTIC.                                    *
1320 45 *  15    CALL DIVIDE_COMPARE (64_REG(1), 32_REG(1))                                                               *
1321 *  16    COUNT := 31                                                                                                  *
1322 *  17    DO WHILE COUNT > 0                                                                                           *
1323 *  18        64_REG(1) := 64_REG(1) .LL. 1                                                                            *
1324 45 *  19        CALL DIVIDE_COMPARE (64_REG(1), 32_REG(1))                                                           *
1325 *  20        COUNT := COUNT-1                                                                                         *
1326 *  21    ENDDO ..END WHILE.                                                                                           *
1327 *  22    64_REG := 64_REG SHIFTED LEFT CIRCULAR 32 BITS                                                               *
1328 *  23    ..64_REG CAN NOW SHARE TERMINATION LOGIC WITH OTHER FLOATING POINT ROUTINES.                                  *
1330 *  24    ..NOTE, 64_REG(63,32) NOW HAS THE QUOTIENT AND 64_REG(31,32) HAS THE REMAINDER.                              *
1332 *  25    IF U(F2)=3 "NO ROUNDING", THEN                                                                               *
1333 *  26        IF 64_REG(1)(63,1)=1 "OVERFLOW OCCURRED", THEN                                                           *
1334 41 *  27            CALL FLOATING_OVERFLOW (64_REG(1), 32_REG(3)) ..UPDATE MANTISSA AND CHARACTERISTIC.               *
1336 *  28        ENDIF                                                                                                    *
1337 *  29    ELSE .."ROUNDING".                                                                                           *
1338 *  30        IF 64_REG(1)(63,1)=1 "OVERFLOW OCCURRED", THEN                                                           *
1339 *  31            ROUND_BIT := 64_REG(1)(32,1) ..SAVE ROUND INDICATOR.                                                 *
1340 41 *  32            CALL FLOATING_OVERFLOW (64_REG(1), 32_REG(3)) ..UPDATE MANTISSA AND CHARACTERISTIC.              *
1342 *  33            IF ROUND_BIT SET, THE ..ROUND UP.                                                                    *
1343 44 *  34                CALL ROUND_UP (64_REG(1)(63,32), 32_REG(3))                                                  *
1344 *  35            ENDIF                                                                                                *
1345 *  36        ELSE ..NO OVERFLOW.                                                                                      *
1346 *  37            64_REG(1)(31,32) := 64_REG(1)(31,32) - (32_REG(1)/2) ..REMAINDER MINUS 1/2 THE DIVISOR.           *
1348 *  38            IF 64_REG(1)(31,32) POSITIVE, THEN ..ROUND UP.                                                       *
1349 44 *  39                CALL ROUND_UP (64_REG(1)(63,32), 32_REG(3))                                                  *
1350 *  40            ENDIF                                                                                                *
1351 *  41        ENDIF                                                                                                    *
1352 *  42    ENDIF                                                                                                        *
1353 *  43    IF SIGN_IND SET, THEN ..CORRECT SIGN.                                                                        *
1354 *  44        64_REG(1)(63,32) := .NOT. 64_REG(1)(63,32)                                                               *
1355 *  45    ENDIF                                                                                                        *
1356 *  46    CALL FLOATINT_POINT_END (64_REG(1)(63,32), 32_REG(3)) ..FINISH THE INSTRUCTION.                              *
1358 *  47    RETURN                                                                                                       *
     *                                                                                                                      *
     ************************************************************************************************************************
```

```
     _XS ..ENTER EXECUTIVE STATE, FMT II F=07 0 A=0

     REF
     PAGE ********************************************************************************************************************
          *                                                                                                                    *
1360      *  1    IF U(I) "INDIRECTION", THEN                                                                                  *
1361   18 *  2        CALL IA_SEQUENCE(DUMMY PARAMETERS) ..OP_READ NOT USED SINCE OPERAND IS NOT FETCHED.                       *
1363      *  3    ENDIF                                                                                                        *
1364   34 *  4    CALL GET_BREG (U(B),32_REG(1))                                                                               *
1365      *  5    CODE := U(SY) + 32_REG(1)(15,16) ..16-BIT ISC TO BE STORED UPON INTERRUPT.                                   *
1367   11 *  6    CALL GENERATE_SYNCHRONOUS_INTERRUPT ("P-"0, ENTER EXECUTIVE STATE, CODE)                                     *
1369      *  7    RETURN;                                                                                                      *
          *                                                                                                                    *
          ********************************************************************************************************************
```

```
     _IPI ..INTERPROCESSOR INTERRUPT, FMT II F=07 G A=1

     REF
     PAGE ********************************************************************************************************************
          *                                                                                                                    *
1371      *  1    IF ASR(19,4)=0 "TASK MODE", THEN                                                                           *
1372   11 *  2        CALL GENERATE_SYNCHRONOUS_INTERRUPT ("P-"1, PRIVILEGED INSTRUCTION VIOLATION)                          *
1374      *  3        ..ABORT INSTRUCTION.                                                                                   *
1375      *  4    ELSE                                                                                                       *
1376      *  5        IF U(I) "INDIRECTION", THEN                                                                            *
1377   18 *  6            CALL IA_SEQUENCE (DUMMY PARAMETERS) ..OP_READ NOT USED SINCE OPERAND IS NOT FETCHED.               *
1379      *  7        ENDIF                                                                                                  *
1380   34 *  8        CALL GET_BREG (U(B), 32_REG(1))                                                                        *
1381      *  9        32_REG(1) := U(SY) + 32_REG(1) ..16-BIT QUANTITY.                                                      *
1382      * 10        IF 32_REG(1)(15,1), THEN                                                                               *
1383      * 11            32_REG(1)(CPS,1) := 0 ..PREVENT INTERRUPTION OF SELF.                                              *
1384      * 12        ENDIF                                                                                                  *
1385      * 13        I := 0                                                                                                 *
1386      * 14        DO WHILE I<8                                                                                           *
1387      * 15            IF 32_REG(1)(I,1), THEN                                                                            *
1388      * 16                SEND INTERPROCESSOR INTERRUPT TO CP(I)                                                         *
1389      * 17            ENDIF                                                                                              *
1390      * 18            I := I+1                                                                                           *
1391      * 19        ENDDO ..END WHILE.                                                                                     *
1392      * 20    ENDIF                                                                                                      *
1393      * 21    RETURN                                                                                                     *
          *                                                                                                                    *
          ********************************************************************************************************************
```

NSWC AN/UYK-7 (CP)
CP INSTRUCTION SET

_AEI (E _PEI) ..ALLOW & PREVENT ENABLE INTERRUPTS, FMT II F=07 1 & 2

```
REF
PAGE ********************************************************************************************************************
     *
     *  1    IF ASR(19,4)=0 "TASK MODE", THEN
1395 *  2        CALL GENERATE_SYNCHRONOUS_INTERRUPT ("P-"1, PRIVILEGED INSTRUCTION VIOLATION)
1396 *  3        ..ABORT INSTRUCTION.
1398 *  4    ELSE
1399 *  5        IF U(I) "INDIRECTION", THEN
1400 *  6            CALL IA_SEQUENCE (DUMMY PARAMETERS) ..OP_READ NOT USED SINCE OPERAND IS NOT FETCHED.
1401 *  7        ENDIF
1403 *  8        CALL GET_BREG(U(B), 32_REG(1))
1404 *  9        32_REG(1) := U(SY) + 32_REG(1) ..COMPUTE LOW 16-BITS OF O_BUS.
1405 * 10        INITIATE AN IOC REQUEST ON IOC(U(A).A.3)
1406 * 11        O_BUS := (U(F).LL.26) + (U(F2).LL.20) + 32REG(I)(15,16) ..FUNCTION CODE AND VALUE TO O_BUS.
1407 * 12        SEND O_BUS TO IOC(U(A).A.3)
1409 * 13        IF REQUEST NOT HONORED .OR. O_BUS NOT RECEIVED WITHIN ACCEPTABLE TIME FRAME, THEN
1410 * 14            CALL GENERATE_SYNCHRONOUS_INTERRUPT ("P-"1, CP-IOC COMMAND RESUME, U(A))
1412 * 15            ..ABORT INSTRUCTION.
1413 * 16        ENDIF
1414 * 17    ENDIF
1415 * 18    RETURN
1416 *
     ********************************************************************************************************************
```

_LIM ..LOAD, ENABLE IOC MONITOR CLOCK, FMT II F=07 3

```
REF
PAGE ********************************************************************************************************************
     *                                                                                                                    *
     *  1    IF ASF(19,4)=0 "TASK MODE", THEN                                                                           *
11   *  2        CALL GENERATE_SYNCHRONOUS_INTERRUPT ("P-"1, PRIVILEGED INSTRUCTION VIOLATION)                          *
     *  3        ..ABORT THE INSTRUCTION.                                                                               *
     *  4    ELSE                                                                                                       *
     *  5        IF U(I) "INDIRECTION", THEN                                                                            *
18   *  6            CALL IA_SEQUENCE (DUMMY PARAMETERS) ..OP_READ NOT USED SINCE OPERAND IS NOT FETCHED.               *
     *  7        ENDIF                                                                                                  *
34   *  8        CALL GET_BREG(U(B), 32_REG(1))                                                                         *
     *  9        32_REG(1) := U(SY) + 32_REG(1) ..COMPUTE LOW 16-BITS OF O_BUS.                                         *
     * 10        INITIATE AN IOC REQUEST ON IOC(U(A).A.3)                                                               *
     * 11        O_BUS := (U(F).LL.Z6) + (U(F2).LL.Z0) +32_REG(1)(15,16) ..FUNCTION CODE AND VALUE TO O_BUS.           *
     * 12        SEND O_BUS TO IOC(U(A).A.3)                                                                            *
     * 13        IF REQUEST NOT HONORED .OR. O_BUS NOT RECEIVED WITHIN ACCEPTABLE TIME FRAME, THEN                      *
11   * 14            CALL GENERATE_SYNCHRONOUS_INTERRUPT ("P-"1, CP-IOC COMMAND RESUME, U(A))                           *
     * 15            ..ABORT INSTRUCTION.                                                                               *
     * 16        ENDIF                                                                                                  *
     * 17    ENDIF                                                                                                      *
     * 18    RETURN                                                                                                     *
     *                                                                                                                    *
     ********************************************************************************************************************
```

_IO ..INITIATE I/O, FMT II F=07 4

```
REF
PAGE ********************************************************************************************************************
     *                                                                                                                    *
     *  1    IF ASF(15,4)=0 "TASK MODE", THEN                                                                           *
11   *  2        CALL GENERATE_SYNCHRONOUS_INTERRUPT ("P-"1, PRIVILEGED INSTRUCTION VIOLATION)                          *
     *  3        ..ABORT THE INSTRUCTION.                                                                               *
     *  4    ELSE                                                                                                       *
     *  5        IF U(I) "INDIRECT ADDRESSING", THEN                                                                    *
18   *  6            CALL IA_SEQUENCE (S_DESIGNATOR, Y, DUMMY PARAMETERS) ..DO CASCADING AS REQUIRED.                   *
     *  7        ELSE ..DEFINE NORMAL BASE (S) REGISTER DESIGNATOR & DISPLACEMENT.                                      *
     *  8            S_DESIGNATOR := U(S) ..DEFINE BASE (S) REGISTER SELECTOR.                                          *
34   *  9            CALL GET_BREG (U(6), 32_REG(I))                                                                    *
     * 10            Y := U(") + 32_REG(I)(15) ..COMPUTE DISPLACEMENT.                                                  *
     * 11        ENDIF                                                                                                  *
32   * 12        CALL ADD_S (S_DESIGNATOR, Y) ..COMPUTE ABSOLUTE ADDRESS.                                               *
     * 13        INITIATE AN I/O REQUEST ON IOC(U(A).A.3)                                                               *
     * 14        O_BUS := U(F).LL.26 + U(F2).LL.20 + Y(17) ..FUNCTION DESIGNATORS & ABSOLUTE ADDRESS.                   *
     * 15        SEND O_BUS TO IOC(U(A).A.3)                                                                            *
     * 16        IF .NOT.(I/O REQUEST HONORED .OR. O_BUS RECEIVED) WITHIN ACCEPTABLE TIME FRAME, THEN                   *
11   * 17            CALL GENERATE_SYNCHRONOUS_INTERRUPT ("P-"1, IOC COMMAND RESUME, U(A).A.3)                          *
     * 18            ..ABORT THIS INSTRUCTION.                                                                          *
     * 19        ENDIF                                                                                                  *
     * 20    ENDIF                                                                                                      *
     * 21    RETURN                                                                                                     *
     *                                                                                                                    *
     ********************************************************************************************************************
```

```
     _18 ..INTERRUPT RETURN, FMT II F=07 5

     REF
     PAGE ********************************************************************************************************************
          *                                                                                                                    *
1468      *  1    IF ASP(19,4)=0 "TASK MODE", THEN                                                                           *
1469   11 *  2        CALL GENERATE_SYNCHRONOUS_INTERRUPT ("P-"1, PRIVILEGED INSTRUCTION VIOLATION)                          *
1471      *  3        ..ABORT THIS INSTRUCTION.                                                                              *
1472      *  4    ELSE                                                                                                       *
1473      *  5        _CLASS := CURRENT INTERRUPT LEVEL (STATE) ..AS INDICATED BY ACTIVE STATUS REGISTER.                    *
1475      *  6        ASR := CMR(0'135'+(4*_CLASS) ..RESTORE ACTIVE STATUS REGISTER.                                         *
1476      *  7        P := CMR(0'137'+(4*_CLASS) ..RESTORE PROGRAM COUNTER.                                                  *
1477      *  8    ENDIF                                                                                                      *
1478      *  9    RETURN                                                                                                     *
          *                                                                                                                    *
          ********************************************************************************************************************
```

```
     _RP ..REPEAT, FMT II F=07 6


     REF
     PAGE ***************************************************************************************************************
          *                                                                                                             *
1480      *  1    .. THE INSTRUCTION SEQUENCE WILL CHECK FOR REPEATABILITY OF THE NEXT                                  *
1481      *  2    ..INSTRUCTION, AND IF REPEATABLE THE REPEAT INDICATOR WILL BE SET.                                    *
1482      *  3    ..SHOULD THE NEXT INSTRUCTION BE FOUND NOT TO BE REPEATABLE (AS PER                                   *
1483      *  4    ..THE REPERTOIRE CARD), THAT INSTRUCTION WILL BE ABORTED AND REPEAT                                   *
1484      *  5    ..MODE TERMINATED.                                                                                    *
1485      *  6    IF U(I) "INDIRECT ADDRESSING", THEN                                                                   *
1486   18 *  7        CALL IA_SEQUENCE (DUMMY PARAMETERS) ..DO CASCADING AS REQUIRED.                                   *
1487      *  8    ENDIF                                                                                                 *
1488      *  9    IF B(7) = 0, THEN ..SKIP NEXT INSTRUCTION.                                                            *
1489      * 10        P(0) := P(0) + 1 ..INCREMENT PROGRAM COUNTER TO THE ADDRESS JUST AFTER THAT OF THE REPEATED   /*
          *                 INSTRUCTION.                                                                              *
1491      * 11    ELSE ..EXECUTE NEXT INSTRUCTION B(7) TIMES OR UNTIL THE CONDITION INDICATED BY U(A).                  *
1493      * 12        SET REPEAT_PENDING INDICATOR                                                                      *
1494      * 13        SAVE U(A) & U(B) IN REPEAT_AB FOR REPEAT SEQUENCE USAGE                                           *
1495      * 14        SAVE U(SY) IN REPEAT_SY                                                                           *
1496      * 15    ENDIF                                                                                                 *
1497      * 16    RETURN                                                                                                *
          *                                                                                                             *
          ***************************************************************************************************************
```

```
     _LA ..LOAD ACCUMULATOR WITH MEMORY CONTENTS, FMT I F=10

     REF
     PAGE ***************************************************************************************************
          *                                                                                                   *
1499   14 *  1    CALL OP_READ (REPEAT_ACCUMULATOR, "DISPLACE"0) ..FETCH OPERAND AS PER K DESIGNATOR.        *
1501   36 *  2    CALL PUT_AREG (U(A), REPEAT_ACCUMULATOR)                                                   *
1502      *  3    RETURN                                                                                     *
          *                                                                                                   *
          ***************************************************************************************************
```

```
     _LXB ..LOAD ACCUMULATOR & INDEX B, FMT 1, F=11

REF
PAGE ********************************************************************************************************************
     *                                                                                                                    *
1504 *  1    SET SPR_PRIVILEGED_INSTRUCTION INDICATOR ..LXB IS A SPECIAL CASE FOR SPR(16,1)=1 & INDIRECT                 /*
     *            ADDRESSING.                                                                                           *
1506   14 *  2    CALL OP_READ (REPEAT_ACCUMULATOR, *DISPLACE=0)                                                        *
1507   36 *  3    CALL PUT_AREG (U(A), REPEAT_ACCUMULATOR)                                                              *
1508   34 *  4    CALL GET_BREG (U(B), 32_REG(1))                                                                       *
1509     *  5    32_REG(1) := 32_REG(1)(15) + 1 ..INCREMENT INDEX (B) REGISTER VALUE.                                  *
1510   37 *  6    CALL PUT_BREG (U(B), 32_REG(1))                                                                       *
1511     *  7    RETURN                                                                                                *
     *                                                                                                                    *
     ********************************************************************************************************************
```

```
     _LDIF ..LOAD DIFFERENCE, FMT I, F=12

     REF
     PAGE ********************************************************************************************************************
          *                                                                                                                    *
1513   14 *  1    CALL OP_READ (REPEAT_ACCUMULATOR, "DISPLACE" 0)                                                             *
1514   33 *  2    CALL GET_AREG (U(A), 32-REG(2))                                                                             *
1515      *  3    REPEAT_ACCUMULATOR := REPEAT_ACCUMULATOR - 32_REG(2) ..ONES COMPLEMENT SUBTRACT.                            *
1517      *  4    UPDATE FIXED POINT OVERFLOW BIT IN ASR (BIT 3)                                                              *
1518   36 *  5    CALL PUT_AREG (U(A)+1, REPEAT_ACCUMULATOR)                                                                  *
1519      *  6    RETURN                                                                                                      *
          *                                                                                                                    *
          ********************************************************************************************************************
```

```
     _ANA ..SUBSTRACT A, FMT I, F=13

     REF
     PAGE ********************************************************************************************************************
          *                                                                                                                    *
1521   14 *  1    CALL OP_READ (32_REG, "DISPLACE"0)                                                                          *
1522   33 *  2    CALL GET_AREG (U(A), REPEAT_ACCUMULATOR)                                                                    *
1523      *  3    REPEAT_ACCUMULATOR := REPEAT_ACCUMULATOR - 32_REG(2) ..ONES COMPLEMENT SUBTRACT.                            *
1525      *  4    UPDATE FIXED POINT OVERFLOW BIT IN ASR (BIT 3)                                                              *
1526   36 *  5    CALL PUT_AREG (U(A), REPEAT_ACCUMULATOR)                                                                    *
1527      *  6    RETURN                                                                                                      *
          *                                                                                                                    *
          ********************************************************************************************************************
```

_AA ..ADD A, FMT I, F=14

```
REF
PAGE ********************************************************************************************************************************
     *                                                                                                                              *
1529   14 *  1    CALL OP_READ (32_REG(2), "DISPLACE" 0)                                                                           *
1530   33 *  2    CALL GET_AREG (U(A), REPEAT_ACCUMULATOR)                                                                         *
1531      *  3    REPEAT_ACCUMULATOR := REPEAT_ACCUMULATOR + 32_REG(2) ..ONES COMPLEMENT ADDITION.                                 *
1533      *  4    UPDATE FIXED POINT OVERFLOW BIT IN ASR (BIT 3)                                                                   *
1534   36 *  5    CALL PUT_AREG (U(A), REPEAT_ACCUMULATOR)                                                                         *
1535      *  6    RETURN                                                                                                           *
     *                                                                                                                              *
     ********************************************************************************************************************************
```

```
     _LSUM ..LOAD SUM, FMT I, F=15

     REF
     PAGE ********************************************************************************************************************
          *                                                                                                                    *
1537   14 *  1    CALL CP_READ (32_REG(2), "DISPLACE" 0)                                                                      *
1538   33 *  2    CALL GET_AREG (U(A), REPEAT_ACCUMULATOR)                                                                    *
1539      *  3    REPEAT_ACCUMULATOR := REPEAT_ACCUMULATOR + 32_REG(2) ..ONES COMPLEMENT ADDITION.                            *
1541      *  4    UPDATE FIXED POINT OVERFLOW BIT IN ASR (BIT 3)                                                              *
1542   36 *  5    CALL PUT_AREG (U(A)+1, REPEAT_ACCUMULATOR)                                                                  *
1543      *  6    RETURN                                                                                                      *
          *                                                                                                                    *
          ********************************************************************************************************************
```

```
     _LNA ..LOAD NEGATIVE, FMT I, F=16

REF
PAGE ***************************************************************************************************************
     *                                                                                                             *
1545   14 *  1    CALL OP_READ (REPEAT_ACCUMULATOR, "DISPLACE" 0)                                                   *
1546      *  2    REPEAT_ACCUMULATOR := - REPEAT_ACCUMULATOR ..ONES COMPLEMENT.                                     *
1547   36 *  3    CALL PUT_AREG (U(A), REPEAT_ACCUMULATOR)                                                          *
1548      *  4    RETURN                                                                                            *
     *                                                                                                             *
     ***************************************************************************************************************
```

```
     _LM ..LOAD MAGNITUDE, FMT I, F=17

     REF
     PAGE ********************************************************************************************************************
          *                                                                                                                    *
1550   14 *  1    CALL OP_READ (REPEAT_ACCUMULATOP, "DISPLACE" 0)                                                             *
1551      *  2    IF REPEAT_ACCUMULATOR < 0, THEN                                                                             *
1552      *  3        REPEAT_ACCUMULATOR := - REPEAT_ACCUMULATOR .. ONES COMPLEMENT.                                          *
1553      *  4    ENDIF                                                                                                       *
1554   36 *  5    CALL PUT_AREG(UCA), REPEAT_ACCUMULATOR)                                                                     *
1555      *  6    RETURN                                                                                                      *
          *                                                                                                                    *
          ********************************************************************************************************************
```

```
     _L8 ..LOAD B, FNT I F=20

     REF
     PAGE ********************************************************************************
          *                                                                              *
1557      *  1    .. Y -> B(U(A))(15)                                                    *
1558   14 *  2    CALL OP_READ(32_REG(1), "DISPLACE"0)                                   *
1559   34 *  3    CALL GET_BREG(U(A), 32_REG(2))                                         *
1560      *  4    32_REG(1)(19,3) := 32_REG(2)(19,3)                                     *
1561   37 *  5    CALL PUT_BREG(U(A), 32_REG(1))                                         *
1562      *  6    RETURN                                                                 *
          *                                                                              *
          ********************************************************************************
```

_AB ..ADD B, FMT 1 F = 21

```
REF
PAGE ********************************************************************************************************************
     *                                                                                                                    *
1564 *  1    .. B(U(A))(15) + Y -> B(U(A))(15)                                                                           *
1565  14 *  2    CALL CP_READ(32_REG(1), "DISPLACE"0)                                                                    *
1566  34 *  3    CALL GET_BREG(U(A), 32_REG(2))                                                                          *
1567     *  4    32_REG(3) := REPEAT_ACCUMULATOR(15)                                                                     *
1568     *  5    32_REG(3) := 32_REG(3) + 32_REG(1) ..ONES COMPLEMENT ADDITION.                                          *
1569     *  6    REPEAT_ACCUMULATOR(15) := 32_REG(3)                                                                     *
1570  37 *  7    CALL PUT_BREG(U(A), REPEAT_ACCUMULATOR)                                                                 *
1571     *  8    RETURN                                                                                                  *
     *                                                                                                                    *
     ********************************************************************************************************************
```

```
     _ANB ..SUBTRACT B, FMT I, F=22

REF
PAGE ********************************************************************************************************************************
     *
1573 *  1    .. B(U(A))(15)-Y -> B(U(A))(15)                                                                                        *
1574   14 *  2    CALL OP_READ (32_REG(1), "DISPLACE"0)                                                                             *
1575   34 *  3    CALL GET_BREG (U(A), REPEAT_ACCUMULATOR)                                                                          *
1576 *  4    32_REG(3) := REPEAT_ACCUMULATOR(15)                                                                                    *
1577 *  5    32_REG(3) := 32_REG(3) - 32_REG(1) ..ONES COMPLEMENT SUBTRACTION.                                                      *
1578 *  6    REPEAT_ACCUMULATOR(15) := 32_REG(3)                                                                                    *
1579   37 *  7    CALL PUT_BREG(U(A), REPEAT_ACCUMULATOR)                                                                           *
1580 *  8    RETURN                                                                                                                 *
     *                                                                                                                             *
     ********************************************************************************************************************************
```

```
_SB ..STORE B FMT I, F=23

REF
PAGE ************************************************************************************************************************
     *                                                                                                                      *
1582 *  1    .. B(U(A))(15) -> Y                                                                                            *
1583  34 *  2    CALL GET_BREG(U(A), REPEAT_ACCUMULATOR)                                                                    *
1584 *  3    REPEAT_ACCUMULATOR := REPEAT_ACCUMULATOR(15)                                                                   *
1585  15 *  4    CALL OP_STORE(REPEAT_ACCUMULATOR, "DISPLACE" 0) ..REPEAT_ACCUMULATOR -> MEMORY.                            *
1587 *  5    RETURN                                                                                                         *
     *                                                                                                                      *
     ************************************************************************************************************************
```

NSWC AN/UYK-7 (CP)
CP INSTRUCTION SET

_SA ..STORE A FMT I, F=24

```
      REF
      PAGE ********************************************************************************************************************
           *                                                                                                                    *
           *                                                                                                                    *
1589       *  1     .. A(U(A)) -> Y                                                                                           *
1590    33 *  2     CALL GET_AREG(U(A), REPEAT_ACCUMULATOR)                                                                   *
1591    15 *  3     CALL OP_STORE(REPEAT_ACCUMULATOR, "DISPLACE" 0)                                                           *
1592       *  4     RETURN                                                                                                    *
           *                                                                                                                    *
           ********************************************************************************************************************
```

_SXB ..STORE A & INDEX B FMT I, F=25

```
REF
PAGE ********************************************************************************************************************
     *                                                                                                                    *
     *  1    .. A(U(A)) -> Y, B(U(B)) := B(U(B))+1                                                                        *
1594 *  2    SET SPR_PRIVILEGED_INSTRUCTION INDICATOR                                                                     *
1595 *  3    CALL _SA                                                                                                     *
1596 88 *  4    CALL GET_BREG(U(B), 32_REG(1))                                                                           *
1597 34 *  5    32_REG(1)(15) := 32_REG(1)(15)+1                                                                         *
1598 *  6    CALL PUT_BREG(U(B), 32_REG(1))                                                                               *
1599 37 *  7    RETURN                                                                                                       *
1600 *                                                                                                                    *
     ********************************************************************************************************************
```

```
_SNA ..STORE NEGATIVE FMT I, F=26

REF
PAGE ********************************************************************************************************************************
     *
1602 *  1    .. (.NOT.A(U(A))) -> Y                                                                                                   *
1603  33 *  2    CALL GET_AREG(U(A), REPEAT_ACCUMULATOR)                                                                               *
1604 *  3    REPEAT_ACCUMULATOR := .NOT. REPEAT_ACCUMULATOR                                                                           *
1605  15 *  4    CALL OP_STORE(REPEAT_ACCUMULATOR, "DISPLACE=0)                                                                       *
1606 *  5    RETURN                                                                                                                   *
     *                                                                                                                               *
     ********************************************************************************************************************************
```

```
     _SM ..STORE MAGNITUDE FMT I, F=27

REF
PAGE ********************************************************************************************************************
     *                                                                                                                  *
1608   33 *  1    CALL GET_AREG(U(A),REPEAT_ACCUMULATOR)                                                               *
1609      *  2    IF REPEAT_ACCUMULATOR(31,1) = 1, THEN                                                                *
1610      *  3        REPEAT_ACCUMULATOR != .NOT. REPEAT_ACCUMULATOR                                                   *
1611      *  4    ENDIF                                                                                                *
1612   15 *  5    CALL OP_STORE(REPEAT_ACCUMULATOR, "DISPLACE"0)                                                       *
1613      *  6    RETURN                                                                                               *
     *                                                                                                                  *
     ********************************************************************************************************************
```

```
     _BZ (_BS) ..CLEAR & SET BIT, FMT I, F=32 & F=33

     REF
     PAGE ********************************************************************************************************************
          *
1615      *  1    32_REG(2) := U(AK) ..SAVE K FIELD & FORCE FULL WORD.                                                        *
1616      *  2    U(K) := FULL WORD TYPE                                                                                      *
1617   14 *  3    CALL OP_READ(REPEAT_ACCUMULATOR, "DISPLACE"0)                                                               *
1618      *  4    IF U(F) EVEN, THEN ..CLEAR BIT INSTRUCTION.                                                                 *
1619      *  5        REPEAT_ACCUMULATOR(32_REG(2)(4),1) := 0                                                                 *
1620      *  6    ELSE ..SET BIT INSTRUCTION.                                                                                 *
1621      *  7        REPEAT_ACCUMULATOR(32_REG(2)(4),1) := 1                                                                 *
1622      *  8    ENDIF                                                                                                       *
1623      *  9    CALL_REPLACE(REPEAT_ACCUMULATOR) ..UPDATE IN MEMORY.                                                        *
1624      * 10    U(K) := 32_REG(2)(2)                                                                                        *
1625      * 11    RETURN                                                                                                      *
          *                                                                                                                   *
          *********************************************************************************************************************
```

_RA (& _RAN) ..REPLACE ADD (SUBTRACT), FMT I, F=34 & F=36

```
REF
PAGE ********************************************************************************************************************
     *
1627   14 *  1    CALL OP_READ (REPEAT_ACCUMULATOR, "DISPLACE"0) ..FETCH OPERAND (Y) UNDER K DESIGNATOR CONTROL.
1629   33 *  2    CALL GET_AREG (U(A), 32_REG(1))
1630      *  3    IF U(F)=0'34' "REPLACE ADD", THEN
1631      *  4        REPEAT_ACCUMULATOR := REPEAT_ACCUMULATOR + 32_REG(1) ..ONES COMPLEMENT ADDITION.
1633      *  5    ELSE ..REPLACE SUBTRACT (F=36).
1634      *  6        REPEAT_ACCUMULATOR := REPEAT_ACCUMULATOR - 32_REG(1) ..ONES COMPLEMENT SUBTRACTION.
1636      *  7    ENDIF
1637      *  8    UPDATE FIXED POINT OVERFLOW (ASR(3,1))
1638      *  9    CALL UPDATE_REPLACE (REPEAT_ACCUMULATOR, U(A)+1)
1639      * 10    RETURN
     *
     ********************************************************************************************************************
```

_RI (& _RD) ..REPLACE INCREMENT (DECREMENT), FMT I, F=35 & F=37

```
REF
PAGE ********************************************************************************************************************************
     *
1641   14 *  1    CALL OP_READ (REPEAT_ACCUMULATOR, "DISPLACE"0)                                                                     *
1642      *  2    IF U(F)="0'35' "REPLACE INCREMENT", THEN                                                                           *
1643      *  3        REPEAT_ACCUMULATOR := REPEAT_ACCUMULATOR + 1 ..ONES COMPLEMENT.                                                *
1644      *  4    ELSE .."REPLACE DECREMENT".                                                                                        *
1645      *  5        REPEAT_ACCUMULATOR := REPEAT_ACCUMULATOR - 1 ..ONES COMPLEMENT.                                                *
1646      *  6    ENDIF                                                                                                              *
1647      *  7    UPDATE FIXED POINT OVERFLOW (ASR(3,1))                                                                             *
1648      *  8    CALL UPDATE_REPLACE (REPEAT_ACCUMULATOR, U(1))                                                                     *
1649      *  9    RETURN                                                                                                             *
     *                                                                                                                              *
     ********************************************************************************************************************************
```

```
_* ..MULTIPLY A, FMT I, F=4C

REF
PAGE ********************************************************************************************************************
     *                                                                                                                    *
1651 *  1    .. A(U(A))*Y -> A(U(A)+1) & A(U(A))                                                                         *
1652   33 *  2    CALL GET_AREG (U(A), 32_REG(2)) ..GET MULTIPLICAND.                                                     *
1653   14 *  3    CALL CP_READ (32_REG(1), "DISPLACE"0) ..GET MULTIPLIER.                                                 *
1654      *  4    SIGN_INDICATOR := 32_REG(1)(31,1) .XOR. 32_REG(2)(31,1) ..REMEMBER SIGN FOR RESULT.                     *
1656   96 *  5    CALL DO_MULTIPLY ..DO ACTUAL MULTIPLICATION.                                                            *
1657      *  6    IF SIGN_INDICATOR "NEGATIVE", THEN ..NEGATE ANSWER.                                                     *
1658      *  7        REPEAT_ACCUMULATOR!=.NOT.REPEAT_ACCUMULATOR; 32_REG(1)!=.NOT.32_REG(1)                              *
1659      *  8    ENDIF                                                                                                   *
1660   36 *  9    CALL PUT_AREG (U(A)+1, REPEAT_ACCUMULATOR) ..STORE MOST SIGNIFICANT BITS.                               *
1662   36 * 10    CALL PUT_AREG (U(A), 32_REG(1)) ..STORE LEAST SIGNIFICANT BITS.                                         *
1663      * 11    RETURN                                                                                                  *
     *                                                                                                                    *
     ********************************************************************************************************************
```

```
      DO_MULTIPLY

      REF
      PAGE ************************************************************************************************************************
           *                                                                                                                      *
1665       *  1    .. MULTIPLY 32_REG(2) BY 32_REG(1) WITH RESULT IN REPEAT_ACCUMULATOR & 32_REG(1),                            *
1667       *  2    ..USING THE UYK-7 ALGORITHM (AS PER "UYK-7 LEARNER'S GUIDE (JAN 1973) PAGE 425).                             *
1669       *  3    FORCE 32_REG(1) POSITIVE & 32_REG(2) NEGATIVE USING ONES COMPLEMENT OPERATIONS.                               *
1671       *  4    REPEAT_ACCUMULATOR := 0; MB_CARRY := 0 "OFF"; COUNT := 16                                                     *
1672       *  5    DO WHILE COUNT>0                                                                                              *
1673       *  6        DO CASE 32_REG(1)(1,2)                                                                                    *
1674       *  7            \0\                                                                                                   *
1675       *  8            IF MB_CARRY, THEN                                                                                     *
1676       *  9                REPEAT_ACCUMULATOR := REPEAT_ACCUMULATOR-32_REG(2); MB_CARRY := 0                                 *
1677       * 10                ..ONES COMPLEMENT SUBTRACT, END AROUND BORROW FORCED.                                             *
1678       * 11            ENDIF                                                                                                 *
1679       * 12            32_REG(1):=32_REG(1).RL.2; 32_REG(1)(31,2):=REPEAT_ACCUMULATOR(1,2)                                   *
1680       * 13            REPEAT_ACCUMULATOR := REPEAT_ACCUMULATOR.RL.2                                                         *
1681       * 14            \1\                                                                                                   *
1682       * 15            IF MB_CARRY, THEN                                                                                     *
1683       * 16                REPEAT_ACCUMULATOR:=REPEAT_ACCUMULATOR-(32_REG(2).LC.1); MB_CARRY:=0                              *
1684       * 17                ..ONES COMPLEMENT SUBTRACT, END AROUND BORROW FORCED.                                             *
1685       * 18            ELSE                                                                                                  *
1686       * 19                REPEAT_ACCUMULATOR := REPEAT_ACCUMULATOR-(32_REG(2)) ..ONES COMPLEMENT SUBTRACT.                  *
1688       * 20            ENDIF                                                                                                 *
1689       * 21            32_REG(1):=32_REG(1).RL.2; 32_REG(1)(31,2):=REPEAT_ACCUMULATOR(1,2)                                   *
1690       * 22            REPEAT_ACCUMULATOR := REPEAT_ACCUMULATOR.RL.2                                                         *
1691       * 23            \2\                                                                                                   *
1692       * 24            IF MB_CARRY, THEN                                                                                     *
1693       * 25                ACCUMULATOR_SIGN := REPEAT_ACCUMULATOR(31,1)                                                      *
1694       * 26                REPEAT_ACCUMULATOR := REPEAT_ACCUMULATOR+32_REG(2) ..ONES COMPLEMENT ADD.                         *
1696       * 27                32_REG(1):=32_REG(1).RL.2; 32_REG(1)(31,2):=REPEAT_ACCUMULATOR(1,2)                               *
1697       * 28                REPEAT_ACCUMULATOR := REPEAT_ACCUMULATOR.RL.2                                                     *
1698       * 29                IF ACCUMULATOR_SIGN "NEGATIVE", THEN                                                              *
1699       * 30                    REPEAT_ACCUMULATOR(31,2) := B'11' ..SET TWO HIGH BITS ON.                                     *
1700       * 31                ENDIF                                                                                             *
1701       * 32            ELSE                                                                                                  *
1702       * 33                REPEAT_ACCUMULATOR := REPEAT_ACCUMULATOR-(32_REG(2).LC.1)                                         *
1703       * 34                ONES COMPLEMENT SUBTRACT, END AROUND BORROW FORCED.                                               *
1704       * 35                SUBTRACT END AROUND BORROW FORCED.                                                                *
1705       * 36                32_REG(1):=32_REG(1).RL.2; 32_REG(1)(31,2):=REPEAT_ACCUMULATOR(1,2)                               *
1706       * 37                REPEAT_ACCUMULATOR := REPEAT_ACCUMULATOR.RL.2                                                     *
1707       * 38                IF .NOT.(END AROUND BORROW), THEN                                                                 *
1708       * 39                    REPEAT_ACCUMULATOR(31,2):= B'01' ..FORCE HIGH BITS TO 0 & 1.                                  *
1709       * 40                ENDIF                                                                                             *
1710       * 41            ENDIF                                                                                                 *
1711       * 42            \3\                                                                                                   *
1712       * 43            IF MB_CARRY, THEN                                                                                     *
1713       * 44                32_REG(1):=32_REG(1).RL.2; 32_REG(1)(31,2):=REPEAT_ACCUMULATOR(1,2)                               *
1714       * 45                REPEAT_ACCUMULATOR := REPEAT_ACCUMULATOR.RL.2                                                     *
1715       * 46            ELSE                                                                                                  *
1716       * 47                MB_CARRY := 1 "ON"                                                                                *
1717       * 48                REPEAT_ACCUMULATOR := REPEAT_ACCUMULATOR+32_REG(2) ..TWOS COMPLEMENT ADD.                         *
1719       * 49                32_REG(1):=32_REG(1).RL.2; 32_REG(1)(31,2):=REPEAT_ACCUMULATOR(1,2)                               *
1720       * 50                REPEAT_ACCUMULATOR := REPEAT_ACCUMULATOR.RL.2                                                     *
1721       * 51            ENDIF                                                                                                 *
1722       * 52        ENDDO ..END CASE.                                                                                         *
1723       * 53        COUNT := COUNT-1                                                                                          *
```

```
     DO_DIVIDE (REF 64_REG, 32_REG)

REF
PAGE ********************************************************************************************************************
     *                                                                                                                  *
1727 *  1    .. DIVIDE THE CONTENTS OF 64_REG BY THE CONTENTS OF 32_REG WITH THE QUOTIENT LEFT IN                       *
1729 *  2    ..64_REG(31,32) AND THE REMAINDER LEFT IN 64_REG(63,32). THIS USES THE UYK-7 ALGORITHM                    *
1731 *  3    ..AS FOUND IN THE UYK-7 LEARNER'S GUIDE (JAN 1973) PAGES 435-442.                                         *
1732 *  4    .. (1) 64_REG - 64-BIT REFERENCE REGISTER.                                                                *
1733 *  5    .. (2) 32_REG - 32-BIT DIVISOR.                                                                           *
1734 *  6    IF 32_REG = -0, THEN                                                                                      *
1735 *  7        32_REG = 0 ..NEGATIVE ZERO IS A SPECIAL CASE.                                                         *
1736 *  8    ENDIF                                                                                                     *
1737 *  9    SIGN_IND := 64_REG(63,1).XOR.32_REG(31,1) ..SIGN CORRECTION FLAG.                                         *
1738 *  10   SIGN_DIVIDEND := 64_REG(63,1) ..SAVE SIGN OF ORIGINAL DIVIDEND.                                           *
1739 *  11   FORCE 64_REG "DIVIDEND" AND 32_REG "DIVISOR" POSITIVE (ONES COMPLEMENT)                                   *
1740 *  12   COUNT := 32                                                                                               *
1741 *  13   DO WHILE COUNT >0                                                                                         *
1742 *  14       64_REG := 64_REG.LL.1                                                                                 *
1743   45 *  15       CALL DIVIDE_COMPARE(64_REG,32_REG)                                                            *
1744 *  16       COUNT := COUNT-1                                                                                      *
1745 *  17   ENDDO ..END WHILE.                                                                                        *
1746 *  18   IF 64_REG(31,1) "QUOTIENT NEGATIVE", THEN ..OVERFLOW HAS OCCURRED.                                         *
1748 *  19       ASR(3,1) "OVERFLOW" := 1                                                                              *
1749 *  20   ELSE                                                                                                      *
1750 *  21       ASR(3,1) "OVERFLOW" := 0                                                                              *
1751 *  22   ENDIF                                                                                                     *
1752 *  23   IF SIGN_IND, THEN                                                                                         *
1753 *  24       64_REG(31,32) := .NOT.64_REG(31,32) ..CORRECT SIGN OF QUOTIENT.                                       *
1754 *  25   ENDIF                                                                                                     *
1755 *  26   IF SIGN_DIVIDEND, THEN                                                                                    *
1756 *  27       64_REG(63,32) := .NOT.64_REG(63,32) ..MAKE REMAINDER THE SAME SIGN AS THE ORIGINAL DIVIDEND.          *
1758 *  28   ENDIF                                                                                                     *
1759 *  29   RETURN                                                                                                    *
     *                                                                                                                  *
     ********************************************************************************************************************
```

```
     _D ..DIVIDE A FMT I, F=41

     REF
     PAGE ********************************************************************************************************************
          *                                                                                                                    *
1761      *  1    .. (A(U(A+1)),A(U(A))) / Y -> A(U(A)); REMAINDER -> A(U(A+1))                                               *
1762   14 *  2    CALL OP_READ(32_REG(1),"DISPLACE"0) ..GET DIVISOR.                                                          *
1763   33 *  3    CALL GET_AREG(U(A),64_REG(1)(31,32)) ..GET 64-BIT DIVIDEND.                                                 *
1764   33 *  4    CALL GET_AREG(U(A+1),64_REG(1)(63,32))                                                                      *
1765   97 *  5    CALL DO_DIVIDE (64_REG(1), 32_REG(1)) ..DO THE ACTUAL DIVISION.                                             *
1766   36 *  6    CALL PUT_AREG(U(A+1),64_REG(1)(63,32)) ..STORE REMAINDER.                                                   *
1767   36 *  7    CALL PUT_AREG(U(A),64_REG(1)(31,32)) .. STORE QUOTIENT.                                                     *
1768      *  8    RETURN                                                                                                      *
          *                                                                                                                    *
          **********************************************************************************************************************
```

```
_BC ..COMPARE BIT TO ZERO, FMT I F=42

REF
PAGE ********************************************************************************************************************
     *                                                                                                                    *
1770 *  1    J:= U(AK)(4) ..IGNORE BIT 25 => BIT # IS MODULO 32.                                                          *
1771 *  2    U(K) := 3 ..FORCE FULL WORD TYPE OPERAND FETCH.                                                              *
1772 *  3    CALL OP=READ(32_REG(1),"DISPLACEMENT")                                                                       *
1773 *  4    IF 32_REG(1)(J,1) = 0 "BIT CLEAR", THEN                                                                      *
1774 *  5        ASR(2,1) := 1 ..INDICATE EQUAL.                                                                          *
1775 *  6    ELSE ..BIT SET.                                                                                              *
1776 *  7        ASR(2,1) := 0 ..INDICATE NOT EQUAL.                                                                      *
1777 *  8    ENDIF                                                                                                        *
1778 *  9    U(AK)(4) := J ..RESTORE THE ORIGINAL AK FIELD.                                                               *
1779 * 10    RETURN                                                                                                       *
     *                                                                                                                    *
     ********************************************************************************************************************
```

_CXI ..COMPARE INDEX INCREMENT, FMT I F=43

```
REF
PAGE ********************************************************************************************************************
     *                                                                                                                    *
1781   34 *  1    CALL GET_BREG(U{A},32_REG(1))                                                                          *
1782   14 *  2    CALL CP_READ(32_REG(2),"DISPLACEMENT"0)                                                                *
1783      *  3    IF 32_REG(1){15,16} >= 32_REG(2), THEN                                                                 *
1784      *  4        ASR{C} := 1 ..INDICATE OUTSIDE LIMITS.                                                             *
1785      *  5        32_REG(1){15,16} := 0 ..CLEAR THE INDEX REGISTER.                                                  *
1786      *  6    ELSE                                                                                                   *
1787      *  7        ASR{C} := 0 ..INDICATE WITHIN LIMITS.                                                              *
1788      *  8        32_REG(1){15,16} := 32_REG(1){15,16} + 1 ..INCREMENT THE INDEX REGISTER.                           *
1789      *  9    ENDIF                                                                                                  *
1790   37 * 10    CALL PUT_BREG(U{A},32_REG(1))                                                                          *
1791      * 11    RETURN                                                                                                 *
     *                                                                                                                    *
     ********************************************************************************************************************
```

```
_C ..COMPARE, FMT I, F=44

REF
PAGE ********************************************************************************************************************************
     *                                                                                                                              *
1793 *  1    .. COMPARE A(U(A)) : OPERAND. SET COMPARE DESIGNATORS IN ASR.                                                          *
1794   33 *  2    CALL GET_AREG (U(A), 32_REG(1))                                                                                   *
1795   14 *  3    CALL OP_READ (32_REG(2), "DISPLACE"0)                                                                             *
1796   29 *  4    CALL SET_CD1 (32_REG(1), 32_REG(2))                                                                               *
1797 *  5    RETURN                                                                                                                 *
     *                                                                                                                              *
     ********************************************************************************************************************************
```

```
_CL ..COMPARE LIMITS, FMT I, F=45

REF
PAGE ***************************************************************************************************************
     *                                                                                                             *
1799 *  1    .. A(U(A)), A(U(A)+1) : OPERAND. SET COMPARE DESIGNATORS IN ASR.                                       *
1800   33 *  2    CALL GET_AREG (U(A), 32_REG(2))                                                                 *
1801   33 *  3    CALL GET_AREG (U(A)+1, 32_REG(3))                                                               *
1802   14 *  4    CALL OP_READ (32_REG(1), *DISPLACE=0)                                                           *
1803   30 *  5    CALL SET_CD2 (32_REG(1), 32_REG( ), 32_REG(3))                                                  *
1804 *  6    RETURN                                                                                                  *
     *                                                                                                             *
     ***************************************************************************************************************
```

```
     _CM ..COMPARE MASKED, FMT I, F=46

     REF
     PAGE ********************************************************************************************************************
          *                                                                                                                    *
1606      *  1    .. A(U(A)+1): = (A(U(A)).AND.OPERAND). SET COMPARE DESIGNATORS IN ASR.                                       *
1607   33 *  2    CALL GET_AREG (U(A)+1, 32_REG(1))                                                                            *
1608   33 *  3    CALL GET_AREG (U(A), 32_REG(2))                                                                              *
1609   14 *  4    CALL OP_READ (32_REG(3), "DISPLACE"0)                                                                        *
1610      *  5    32_REG(2) := 32_REG(2) .A. 32_REG(3)                                                                         *
1611   29 *  6    CALL SET_CD1 (32_REG(1), 32_REG(2))                                                                          *
1612      *  7    RETURN                                                                                                       *
          *                                                                                                                    *
          ********************************************************************************************************************
```

```
      CG ..COMPARE GATED, FMT I, F=47

      REF
      PAGE ********************************************************************************************************************
           *                                                                                                                    *
1814       *  1    .. ABS(OPERAND - A(U(A))) : A(U(A)+1). SET COMPARE DESIGNATORS IN ASR.                                       *
1815    14 *  2    CALL OP_READ (32_REG(1), "DISPLACE"0)                                                                        *
1816    33 *  3    CALL GET_AREG (U(A), 32_REG(3))                                                                              *
1817       *  4    32_REG(1) := ABSOLUTE VALUE OF (32_REG(1) - 32_REG(3))                                                       *
1818    33 *  5    CALL GET_AREG (U(A)+1, 32_REG(2))                                                                            *
1819    29 *  6    CALL SET_CD1 (32_REG(1), 32_REG(2))                                                                          *
1820       *  7    RETURN                                                                                                       *
           *                                                                                                                    *
           ********************************************************************************************************************
```

_JEP (_JOP) ..JUMP EVEN (ODD) PARITY, FMT II, F=50 0 & F=50 1

```
REF
PAGE ********************************************************************************************************************
     *                                                                                                                    *
1822 *  1    SET SPR_PRIVILEGED_INSTRUCTION INDICATOR ..INDICATE PRIVILEGED IF SPR(16,1) SET AND U(I) SET.                *
1824 * 13 2  CALL JUMP_ADDRESS ("DISPLACE"0, OPERAND_S, OPERAND_DISPLACEMENT)                                           *
1825 *  3    ..DO OPERAND CHECKS FOR JUMP TYPE INSTRUCTIONS. OPERAND BASE                                               *
1826 *  4    ..REGISTER AND OPERAND DISPLACEMENT ARE RETURNED.                                                          *
1827 * 33 5  CALL GET_AREG (U(A), 32_REG(1))                                                                           *
1828 * 33 6  CALL GET_AREG (U(A)+1, 32_REG(2))                                                                         *
1829 *  7    32_REG(1)) := 32_REG(1) .A. 32_REG(2)                                                                      *
1830 *  8    I := 0                                                                                                     *
1831 *  9    J := 0                                                                                                     *
1832 * 10    DO _WHILE I < 32                                                                                           *
1833 * 11        J := J + 32_REG(1)(I,1)                                                                                *
1834 * 12        I := I + 1                                                                                             *
1835 * 13    ENDDO                                                                                                      *
1836 * 14    J := J .X. U(F2)                                                                                           *
1837 * 15    IF J(0) = 0, THEN ..TEST FOR JOP/ODD OR JEP/EVEN                                                           *
1838 * 23 16     CALL DO_JUMP (OPERAND_S, OPERNAD_DISPLACEMENT)                                                         *
1839 * 17    ENDIF                                                                                                      *
1840 * 18    RETURN                                                                                                     *
     *                                                                                                                    *
     ********************************************************************************************************************
```

```
_DJZ ..JUMP DOUBLE PRECISION ZERO, FMT III F=50 2

REF
PAGE ********************************************************************************************************************
     *
1842 *  1    SET SPR_PRIVILEGED_INSTRUCTION ..INDICATE PRIVILEGED IF SPR(16,1) SET AND U(I) SET.                        *
1844   13 *  2    CALL JUMP_ADDRESS("DISPLACE"D, OPERAND_S, OPERAND_DISPLACEMENT)                                        *
1845     * 3    ..DO OPERAND CHECKS FOR JUMP TYPE INSTRUCTIONS.                                                         *
1846   33 *  4    CALL GET_AREG(U(A),64_REG(1)(31,32))                                                                   *
1847   33 *  5    CALL GET_AREG(U(A)+1,64_REG(1)(63,32))                                                                 *
1848     * 6    IF 64_REG(1) = 0, THEN ..JUMP ON POSITIVE ZERO ONLY!                                                   *
1849   23 *  7        CALL DO_JUMP (OPERAND_S, OPERAND_DISPLACEMENT)                                                     *
1850     * 8    ENDIF                                                                                                  *
1851     * 9    RETURN                                                                                                 *
     *                                                                                                                 *
     *******************************************************************************************************************
```

JUMP ...UMP DOUBLE PRECISION NOT ZERO, FMT III F=50 3

```
REF
PAGE ************************************************************************************************************
     *
1853 *  1    SET SPR_PRIVILEGED_INSTRUCTION ..INDICATE PRIVILEGED IF SPR(16,1) SET AND U(I) SET.
1855  13 *  2    CALL JUMP_ADDRESS("DISPLACE"0, OPERAND_S, OPERAND_DISPLACEMENT)
1856     *  3    ..DO OPERAND CHECKS FOR JUMP TYPE INSTRUCTIONS.
1857  33 *  4    CALL GET_AREG(U(A),64_REG(1)(31,32))
1858  33 *  5    CALL GET_AREG(L(A)+1,64_REG(1)(63,32))
1859     *  6    IF 64_REG(1) <> 0, THEN ..TEST POSITIVE ZERO ONLY!
1860  23 *  7        CALL DO_JUMP (OPERAND_S, OPERAND_DISPLACEMENT)
1861     *  8    ENDIF
1862     *  9    RETURN
     *
     ************************************************************************************************************
```

_F51(_JP, _JN, _JZ, _JNZ) FMT III F=51 0 THRU 51 3

```
REF
PAGE ********************************************************************************************************************
     *                                                                                                                    *
1864 *  1    SET SPR_PRIVILEGED_INSTRUCTION ..INDICATE PRIVILEGED IF SPR(16,1) SET AND U(I) SET.                          *
1866 * 13  2    CALL JUMP_ADDRESS("DISPLACE"0, OPERAND_S, OPERAND_DISPLACEMENT)                                           *
1867 *  3    ..DO OPERAND CHECKS FOR JUMP TYPE INSTRUCTIONS.                                                             *
1868 * 33  4    CALL GET_AREG(U(A),32(REG(1))                                                                           *
1869 *  5    JUMP := FALSE                                                                                               *
1870 *  6    DO _CASE U(F3) OF                                                                                           *
1871 *  7        \0\ IF 32(REG(1)(31,1) = 0 "POSITIVE", THEN JUMP := TRUE ENDIF                                          *
1872 *  8        \1\ IF 32(REG(1)(31,1) = 1 "NEGATIVE", THEN JUMP := TRUE ENDIF                                          *
1873 *  9        \2\ IF 32(REG(1) = 0 "POSITIVE ZERO", THEN JUMP := TRUE ENDIF                                           *
1874 * 10        \3\ IF 32(REG(1) <> 0 "NOT POSITIVE ZERO", THEN JUMP := TRUE ENDIF                                      *
1875 * 11    ENDDO                                                                                                       *
1876 * 12    IF JUMP := TRUE                                                                                             *
1877 * 23 * 13        CALL DO_JUMP (OPERAND_S, OPERAND_DISPLACEMENT)                                                      *
1878 * 14    ENDIF                                                                                                       *
1879 * 15    RETURN                                                                                                      *
     *                                                                                                                    *
     ********************************************************************************************************************
```

NSWC AN/UYK-7 (CP)
CP INSTRUCTION SET

```
      _LBJ ..LOAD B AND JUMP, FMT III, F=52 C

      REF
      PAGE ********************************************************************************************************************
           *                                                                                                                    *
           *  1    SET SPR_PRIVILEGED_INSTRUCTION ..INDICATE PRIVILEGED IF SPR(16,1) SET AND U(I) SET.                          *
1881       *  2    IF U(A) = 0, THEN                                                                                            *
1883       *  3        RETURN ..NO OPERATION.                                                                                   *
1884       *  4    ELSE                                                                                                         *
1885       *  5        32_REG(1)(19,3) := P(S) ..P REGISTER BASE REGISTER DESIGNATOR.                                           *
1886       *  6        32_REG(1)(15,16) := P(D) ..P REGISTER DISPLACEMENT (ALREADY INCREMENTED).                                *
1887       *  6        CALL PUT_BREG(U(A),32(REG(1))                                                                            *
1889    37 *  7        CALL JUMP_ADDRESS("DISPLACE"0, OPERAND_S, OPERAND_DISPLACEMENT)                                          *
1890    13 *  8        ..DO OPERAND CHECKS FOR JUMP TYPE INSTRUCTIONS.                                                          *
1891       *  9        ..DO OPERAND CHECKS FOR JUMP TYPE INSTRUCTIONS.                                                          *
1892    23 * 10        CALL DO_JUMP (OPERAND_S, OPERAND_DISPLACEMENT)                                                           *
1893       * 11    ENDIF                                                                                                        *
1894       * 12    RETURN                                                                                                       *
           *                                                                                                                    *
           ********************************************************************************************************************
```

```
_JBNZ ..INDEX JUMP B, FMT III, F=52 I

REF
PAGE ********************************************************************************************************************************
     *
1896 *  1    SET SPR_PRIVILEGED_INSTRUCTION ..INDICATE PRIVILEGED IF SPR(16,1) SET AND U(I) SET.
1898  34 *  2    CALL GET_BREG(U(A),32(REG(I))
1899     *  3    IF 32(REG(I)(15,16) <> 0, THEN
1900     *  4        32_REG(I)(15,16) := 32_REG(I)(15,16) - 1 ..DECREMENT THE INDEX REGISTER.
1902  37 *  5        CALL PUT_BREG(U(A),32(REG(I))
1903  13 *  6        CALL JUMP_ADDRESS("DISPLACE"0, OPERAND_S, OPERAND_DISPLACEMENT)
1904     *  7        ..DO OPERAND CHECKS FOR JUMP TYPE INSTRUCTIONS.
1905  23 *  8        CALL DO_JUMP (OPERAND_S, OPERAND_DISPLACEMENT)
1906     *  9    ENDIF
1907     * 10    RETURN
     *
     ********************************************************************************************************************************
```

```
     _JS ..JUMP SY & B, FMT III F=52 Z

     REF
     PAGE ********************************************************************************************************
          *                                                                                                      *
1909   34 *  1    CALL GET_BREG(U(5), 32_REG(1))                                                                 *
1910      *  2    P(S) := 32_REG(1)(19,3) ..P(S) GETS NEW BASE REGISTER DESIGNATOR.                              *
1911      *  3    P(D) := 32_REG(1)(15) + U(SY) ..P(D) GETS NEW DISPLACEMENT.                                    *
1912      *  4    32_REG(1) := P(D)                                                                              *
1913   32 *  5    CALL ADD_S(P(S), 32_REG(1)) ..JUMPED TO ADDRESS GOES TO 32_REG(1).                             *
1914   17 *  6    CALL BPR_CHECK(32_REG(1), OPERAND) ..DO OPERAND CHECK ON JUMPED TO ADDRESS.                    *
1916      *  7    RETURN                                                                                         *
          *                                                                                                      *
          ********************************************************************************************************
```

```
_JL ..JUMP LOWER, FMT III, F=52 3

REF
PAGE ***************************************************************************************************************
     *                                                                                                             *
1918 *   1    SET SPR_PRIVILEGED_INSTRUCTION ..INDICATE PRIVILEGED IF SPR(16,1) SET AND U(I) SET.                  *
1920 *   2    .. NOTE: BPR CHECKS & ILLEGAL INSTRUCTION CHECKS ARE DONE IN THIS ROUTINE TO                         *
1922 *   3    ..SUPPLEMENT NORMAL I SEQUENCE HANDLING OF THE LOWER HALF WORD INSTRUCTION.                          *
1924  13 *   4    CALL JUMP_ADDRESS("DISPLACE"0, OPERAND_S, OPERAND_DISPLACEMENT)                                  *
1925 *   5    ..DO OPERAND CHECKS FOR JUMP TYPE INSTRUCTIONS.                                                      *
1926 *   6    P(S) := OPERAND_S ..SET UP BASE (S) DESIGNATOR FOR JUMPED TO INSTRUCTION.                            *
1928 *   7    P(D) := OPERAND_DISPLACEMENT ..SET UP DISPLACEMENT FOR JUMPED TO INSTRUCTION.                        *
1930 *   8    32_REG(1) := P(D) ..COPY THE DISPLACEMENT FOR CHECKS.                                                *
1931 *   9    ASR(15,1) := 1 ..SET LOWER HALF WORD INDICATOR.                                                      *
1932  32 * 10    CALL ADD_S(U(S),32_REG(1)) ..COMPUTE ABSOLUTE ADDRESS OF JUMPED TO INSTRUCTION.                  *
1934  17 * 11    CALL BPR_CHECK(32_REG(1),INSTRUCTION)                                                            *
1935 *  12    ..DO BPR CHECKS, PECULIAR TO LOWER HALF WORD INSTRUCTIONS ARRIVED AT BY JL INSTRUCTION.              *
1937  19 * 13    CALL MEMORY_READ(32_REG(1), 32_REG(2), U(S), INSTRUCTION) ..FETCH INSTRUCTION.                   *
1939 *  14    .. NOTE: THE INITIAL CALL TO OP_READ INSURES THE INSTRUCTION IS AT A                                 *
1940 *  15    ..VALID MEMORY LOCATION, THUS NO INTERRUPT SHOULD OCCUR.                                             *
1941 *  16    IF 32_REG(2)(15,2) <> B'11', THEN ..CHECK FOR VALID HALF-WORD INSTRUCTION CANDIDATE.                 *
1943  12 * 17        CALL GENERATE_SYNCHRONOUS_INTERRUPT("P-"1, OP ILLEGAL INSTRUCTION)                           *
1944 *  18    END                                                                                                  *
1945 *  19    RETURN                                                                                               *
     *                                                                                                             *
     ***************************************************************************************************************
```

_J530 ..JUMP ON ASR OVERFLOW DESIGNATOR (JNF, JOF), FMT III F=53 0

```
REF
PAGE ********************************************************************************
     *
1947 *  1    SET SPR_PRIVILEGED_INSTRUCTION ..INDICATE PRIVILEGED IF SPR(I6,1) SET AND U(I) SET.
1949   13 *  2    CALL JUMP_ADDRESS("DISPLACE"G, OPERAND_S, OPERAND_DISPLACEMENT)
1950 *  3    ..DO OPERAND CHECKS FOR JUMP TYPE INSTRUCTIONS..
1951 *  4    J := ASR(3,1) "OUTSIDE/WITHIN" .X. U(A)
1952 *  5    ASR(3,1) := 0 ..CLEAR ASR OVERFLOW INDICATOR.
1953 *  6    IF J(0,1)=0, THEN
1954   23 *  7        CALL DO_JUMP(OPERAND_S, OPERAND_DISPLACEMENT)
1955 *  8    ENDIF
1956 *  9    RETURN
     *
     ********************************************************************************
```

```
     _J531 ..JUMP ON ASR COMPARE DESIGNATOR, FMT III F=53 1

REF
PAGE ********************************************************************************************************************
     *                                                                                                                    *
1958 *  1    .. THIS ROUTINE EXECUTES THE FOLLOWING INSTRUCTIONS : JNE, JE, JG, JGE, JLT, JLE, JNW, JW.                   *
1960 *  2    SET SPR_PRIVILEGED_INSTRUCTION ..INDICATE PRIVILEGED IF SPR(16,1) SET AND U(I) SET.                         *
1962   13 *  3    CALL JUMP_ADDRESS("DISPLACE"O, OPERAND_S, OPERAND_DISPLACEMENT)                                         *
1963 *  4    ..DO OPERAND CHECKS FOR JUMP TYPE INSTRUCTIONS                                                              *
1964 *  5    _JUMP := FALSE ..INITIALIZE THE JUMP FLAG.                                                                  *
1965 *  6    DO CASE U(A) OF                                                                                             *
1966 *  7        .. \0\ \1\ ..JNE, JE.                                                                                   *
1967 *  8        J := U(A) .X. ASR(2,1) "EQUAL/UNUQUAL"                                                                  *
1968 *  9        IF J(0,1)=0, THEN                                                                                       *
1969 * 10            _JUMP := TRUE                                                                                       *
1970 * 11        ENDIF                                                                                                   *
1971 * 12        .. \2\ ..JG.                                                                                            *
1972 * 13        IF ASR(1,1)=1 "GREATER THAN OR EQUAL" .A. ASR(2,1)=0 "UNEGUAL", THEN                                    *
1973 * 14            _JUMP := TRUE                                                                                       *
1974 * 15        ENDIF                                                                                                   *
1975 * 16        .. \3\ \4\ ..JGE, JLT.                                                                                  *
1976 * 17        J := U(A) .X. ASR(1,1) "GREATER THAN OR EQUAL/LESS THAN"                                                *
1977 * 18        IF J(0,1)=0, THEN                                                                                       *
1978 * 19            _JUMP := TRUE                                                                                       *
1979 * 20        ENDIF                                                                                                   *
1980 * 21        .. \5\ ..JLE.                                                                                           *
1981 * 22        IF ASR(1,1)=0 "LESS THAN" .V. ASR(2,1)=1 "EQUAL", THEN                                                  *
1982 * 23            _JUMP := TRUE                                                                                       *
1983 * 24        ENDIF                                                                                                   *
1984 * 25        .. \6\ \7\ ..JNW, JW.                                                                                   *
1985 * 26        J := U(A) .X. ASR(0,1) "OUTSIDE/WITHIN"                                                                 *
1986 * 27        IF J(0,1)=1, THEN                                                                                       *
1987 * 28            _JUMP := TRUE                                                                                       *
1988 * 29        ENDIF                                                                                                   *
1989 * 30    ENDDO                                                                                                       *
1990 * 31    IF _JUMP = TRUE, THEN                                                                                       *
1991   23 * 32        CALL DO_JUMP(OPERAND_S, OPERAND_DISPLACEMENT)                                                       *
1992 * 33    ENDIF                                                                                                       *
1993 * 34    RETURN                                                                                                      *
     *                                                                                                                    *
     ********************************************************************************************************************
```

```
_J532 ..RETURN JUMPS, FMT III F=53 2

REF
PAGE ***************************************************************************************************
     *                                                                                                   *
1995 *  1    SET SPR_PRIVILEGED_INSTRUCTION ..INDICATE PRIVILEGED IF SPP(16,1) SET AND U(I) SET.        *
1997 *  2    _MATCH := FALSE ..INITIALIZE THE MATCH FLAG.                                               *
1998 *  3    JUMP_STOP := U(A)(2,1) ..HIGH ORDER BIT OF "A" FIELD.                                      *
1999 *  4    IF JUMP_STOP .AND. ASR(19,4)=0 "TASK MODE", THEN                                           *
2000  11 *  5        GENERATE_SYNCHRONOUS_INTERRUPT("P-"1, PRIVILEGED INSTRUCTION VIOLATION)            *
2001 *  6        .. ABORTS THE INSTRUCTION.                                                             *
2002 *  7    ENDIF                                                                                      *
2003 *  8    IF U(A)(1,2), THEN ..CHECK APPROPRIATE JUMP SWITCH.                                        *
2004 *  9        IF JUMP_STOP, THEN                                                                     *
2005 * 10            IF STOP_SWITCH(U(A)) "=1", THEN                                                    *
2006 * 11                _MATCH := TRUE                                                                 *
2007 * 12            ENDIF                                                                              *
2008 * 13        ELSE                                                                                   *
2009 * 14            IF JUMP_SWITCH(U(A)) "=1", THEN                                                    *
2010 * 15                _MATCH := TRUE                                                                 *
2011 * 16            ENDIF                                                                              *
2012 * 17        ENDIF                                                                                  *
2013 * 18    ELSE ..0 -> ALWAYS JUMP                                                                    *
2014 * 19        _MATCH := TRUE                                                                         *
2015 * 20    ENDIF                                                                                      *
2016 * 21    32_REG(1)(31,15) := P(S)                                                                   *
2017 * 22    32_REG(1)(15) := P(0)                                                                      *
2018  15 * 23    CALL OP_STORE(32_REG(1), "DISPLACE"0) ..STORE P.                                       *
2020  13 * 24    CALL JUMP_ADDRESS("DISPLACE"1, OPERAND_S, OPERAND_DISPLACEMENT)                        *
2021 * 25    ..DO OPERAND CHECKS FOR JUMP TYPE INSTRUCTIONS.                                            *
2022 * 26    IF JUMP_STOP .OR. (_MATCH = TRUE), THEN                                                    *
2023  23 * 27        CALL DO_JUMP(OPERAND_S, OPERAND_DISPLACEMENT)                                      *
2024 * 28        IF JUMP_STOP .AND. (_MATCH = TRUE), THEN                                               *
2025 * 29            SUSPEND PROCESSING (INDICATING STOP ACTIVE) UNTIL RESTARTED                        *
2026 * 30        ENDIF                                                                                  *
2027 * 31    ENDIF                                                                                      *
2028 * 32    RETURN                                                                                     *
     *                                                                                                   *
     *****************************************************************************************************
```

J533 ..MANUAL JUMPS, FMT III, F=53 3

```
REF
PAGE ********************************************************************************************************************************
     *
2030 *  1    SET SPR_PRIVILEGED_INSTRUCTION INDICATOR                                                                              *
2031 * 13 2  CALL JUMP_ADDRESS ("DISPLACE"0, OPERAND_S, OPERAND_DISPLACEMENT)                                                      *
2032 *  3    _MATCH := FALSE ..ASSUME NO JUMP.                                                                                     *
2033 *  4    JUMP_STOP := U(A)(2,1) ..I.E. HIGH BIT OF A FIELD.                                                                    *
2034 *  5    IF JUMP_STOP .AND. ASR(19,4)=0 "TASK MODE", THEN                                                                      *
2035 * 11 6      CALL GENERATE_SYNCHRONOUS_INTERRUPT ("P-"1, PRIVILEGED INSTRUCTION VIOLATION)                                     *
2037 *  7        ..ABORT THIS INSTRUCTION.                                                                                         *
2038 *  8    ENDIF                                                                                                                 *
2039 *  9    IF U(A)(1,2), THEN ..CHECK APPROPRIATE JUMP SWITCH.                                                                   *
2040 * 10        IF JUMP_STOP, THEN                                                                                               *
2041 * 11            IF STOP_SWITCH(U(A)) "=1", THEN                                                                              *
2042 * 12                _MATCH := TRUE                                                                                           *
2043 * 13            ENDIF                                                                                                        *
2044 * 14        ELSE                                                                                                             *
2045 * 15            IF JUMP_SWITCH(U(A)) "=1", THEN                                                                              *
2046 * 16                _MATCH := TRUE                                                                                           *
2047 * 17            ENDIF                                                                                                        *
2048 * 18        ENDIF                                                                                                            *
2049 * 19    ELSE ..0 => ALWAYS JUMP.                                                                                             *
2050 * 20        _MATCH := TRUE                                                                                                   *
2051 * 21    ENDIF                                                                                                                *
2052 * 22    IF (JUMP_STOP .OR. (_MATCH=TRUE)), THEN                                                                              *
2053 * 23 23     CALL DO_JUMP (OPERAND_S, OPERAND_DISPLACEMENT)                                                                   *
2054 * 24        IF (JUMP_STOP .AND. (_MATCH=TRUE)), THEN                                                                         *
2055 * 25            SUSPEND PROCESSING (INDICATING STOP ACTIVE) UNTIL RESTARTED                                                  *
2056 * 26        ENDIF                                                                                                            *
2057 * 27    ENDIF                                                                                                                *
2058 * 28    RETURN                                                                                                               *
     *                                                                                                                           *
     ********************************************************************************************************************************
```

LCT (& LCI) ..LOAD CMR, FMT I, F=54 & F=55

```
REF
PAGE ********************************************************************************************************************
     *                                                                                                                    *
2060 *  1    ..TREATED AS A FORMAT III INSTRUCTION IN THAT U(K) DOESN'T AFFECT OPERAND.                                     *
2062 *  2    32_REG(2) := U(AK) ..ISOLATE AND SAVE "AK" FIELD.                                                             *
2063 *  3    IF U(F) ODD, THEN ..INTERRUPT CONTROL MEMORY REFERENCED.                                                      *
2064 *  4        32_REG(2) := 32_REG(2)+0'100' ..DISPLACE TO INTERRUPT ADDRESSES.                                          *
2065 *  5    ENDIF                                                                                                         *
2066 *  6    IF (32_REG(2)>0'17' .OR. REPEAT_IN_PROGRESS) .AND. ASR(19,4)=0 "TASK MODE", THEN                              *
2068 11 *  7        CALL GENERATE_SYNCHRONOUS_INTERRUPT ("P-"1, PRIVILEGED INSTRUCTION VIOLATION)                         *
2070 *  8        ..ABORT THIS INSTRUCTION.                                                                                 *
2071 *  9    ENDIF                                                                                                         *
2072 * 10    U(K) := 3 "FULL WORD"                                                                                         *
2073 14 * 11    CALL OP_READ (REPEAT_ACCUMULATOR, "DISPLACE"0)                                                            *
2074 * 12    CMR(32_REG(2)) := REPEAT_ACCUMULATOR ..UPDATE CONTROL MEMORY CONTENTS.                                        *
2075 * 13    IF REPEAT_IN_PROGRESS, THEN                                                                                   *
2076 * 14        32_REG(2) := 32_REG(2)+1 ..INCREMENT AK FIELD.                                                            *
2077 * 15    ENDIF                                                                                                         *
2078 * 16    U(AK) := 32_REG(2)(5)                                                                                         *
2079 * 17    RETURN                                                                                                        *
     *                                                                                                                    *
     ********************************************************************************************************************
```

```
      SCT (& SCI) ..STORE CMR, FMT I, F=56 & F=57

      REF
      PAGE ********************************************************************************************************************
           *                                                                                                                    *
2081       *  1    ..TREATED AS A FORMAT III INSTRUCTION IN THAT U(K) DOESN'T AFFECT OPERAND.                                   *
2083       *  2    32_REG(2) := U(AK) ..ISOLATE AND SAVE "AK" FIELD.                                                           *
2084       *  3    IF U(F) ODD, THEN ..INTERRUPT CONTROL MEMORY REFERENCED.                                                    *
2085       *  4        32_REG(2) := 32_REG(2)+O'100' ..DISPLACE TO INTERRUPT ADDRESSES.                                        *
2086       *  5    ENDIF                                                                                                       *
2087       *  6    IF (32_REG(2)>O'17' .OR. REPEAT_IN_PROGRESS) .AND. ASR(19,4)=0 "TASK MODE", THEN                            *
2089    11 *  7        CALL GENERATE_SYNCHRONOUS_INTERRUPT ("P-"1, PRIVILEGED INSTRUCTION VIOLATION)                           *
2091       *  8        ..ABORT THIS INSTRUCTION.                                                                               *
2092       *  9    ENDIF                                                                                                       *
2093       * 10    REPEAT_ACCUMULATOR := CMR(32_REG(2)) ..CONTROL MEMORY CONTENTS GOES TO REPEAT_ACCUMULATOR.                  *
2095       * 11    U(K) := 3 "FULL WORD"                                                                                       *
2096    15 * 12    CALL OP_STORE (REPEAT_ACCUMULATOR, "DISPLACE"0)                                                             *
2097       * 13    IF REPEAT_IN_PROGRESS, THEN                                                                                 *
2098       * 14        32_REG(2) := 32_REG(2)+1 ..INCREMENT AK FIELD.                                                          *
2099       * 15    ENDIF                                                                                                       *
2100       * 16    U(AK) := 32_REG(2)(5) ..RESTORE AK FIELD.                                                                   *
2101       * 17    RETURN                                                                                                      *
           *                                                                                                                    *
           ********************************************************************************************************************
```

```
     HSC_60 ..STORE CMR IN A, FRT IV A, F=60

REF
PAGE ********************************************************************************************************************
     *                                                                                                                    *
2103 *  1    ADR := U(AF4) ..EXTRACT CONTROL MEMORY ADDRESS (I.E. 6-BIT "A" & "F4" FIELDS).                               *
2105 *  2    IF U(I) "HSCI", THEN                                                                                         *
2106 *  3        ADR := ADR + O'100' ..DISPLACE TO INTERRUPT ADDRESSES.                                                   *
2107 *  4    ENDIF                                                                                                        *
2108 *  5    IF ADR>=O'20' .AND. ASR(19,4)=0 "TASK MODE", THEN                                                            *
2109 *  6        CALL GENERATE_SYNCHRONOUS_INTERRUPT ("P-"1, PRIVILEGED INSTRUCTION VIOLATION)                            *
2111 *  7        ..ABORT THIS INSTRUCTION.                                                                                *
2112 *  8    ENDIF                                                                                                        *
2113 *  9    CALL HALFWORD_TOGGLE ..CALL AFTER INTERRUPT POSSIBILITY FOR ECP 97A.                                         *
2114 * 10    IF (ADR<=O'137' .AND. ADR>=O'130') .OR. (ADR<=O'57' .AND. ADR>=O'30'), THEN                                  *
2116 * 11        32_REG(1) := 0 ..UNASSIGNED CONTROL MEMORY IS A SOURCE OF ZEROS.                                         *
2117 * 12    ELSE                                                                                                         *
2118 * 13        IF U(AF4)<=O'17' .AND. U(AF4)>=O'11' "INDEX REGISTER REFERENCE", THEN                                    *
2119 * 14            32_REG(1) := CMR(ADR)(15) ..EXTRACT ONLY 16-BITS.                                                    *
2120 * 15        ELSE                                                                                                     *
2121 * 16            IF ADR<=O'77' .AND. ADR>=O'70' "ACTIVE STATUS REGISTER", THEN                                        *
2122 * 17                32_REG(1) := CMR(70) ..ALL 7X REFER TO ASR.                                                      *
2123 * 18            ELSE                                                                                                 *
2124 * 19                IF ADR>=60 .AND. ADR<=67 "BREAKPOINT REGISTER", THEN                                             *
2125 * 20                    32_REG(1) := CMR(60) ..ALL 6X REFER TO BREAKPOINT REGISTER.                                  *
2126 * 21                ELSE ..ALL OTHER REFERENCES.                                                                     *
2127 * 22                    32_REG(1) := CMR(ADR)                                                                        *
2128 * 23                ENDIF                                                                                            *
2129 * 24            ENDIF                                                                                                *
2130 * 25        ENDIF                                                                                                    *
2131 * 26    ENDIF                                                                                                        *
2132   36 * 27    CALL PUT_AREG (U(B), 32_REG(1))                                                                         *
2133 * 28    RETURN                                                                                                       *
     *                                                                                                                    *
     ********************************************************************************************************************
```

HLC_61 ..LOAD CMR FROM A, FMT IV A, F=61

```
REF
PAGE ********************************************************************************************************************
     *                                                                                                                    *
2135 *  1    ADR := U(AF4) ..EXTRACT CONTROL MEMORY ADDRESS (I.E. 6-BIT "A" & "F4" FIELDS).                               *
2137 *  2    IF U(5) "HLCI", THEN                                                                                         *
2138 *  3        ADR := ADR + O'100' ..DISPLACE TO INTERRUPT ADDRESSES.                                                    *
2139 *  4    ENDIF                                                                                                        *
2140 *  5    IF ADR>=O'20' .AND. ASR(19,4)=0 "TASK MODE", THEN                                                            *
2141 *  6        CALL GENERATE_SYNCHRONOUS_INTERRUPT ("P-"1, PRIVILEGED INSTRUCTION VIOLATION)                            *
2143 *  7        ..ABORT THIS INSTRUCTION.                                                                                *
2144 *  8    ENDIF                                                                                                        *
2145 *  9    CALL HALFWORD_TOGGLE ..CALL AFTER INTERRUPT POSSIBILITY FOR ECP 9TA.                                         *
2146 * 10    IF (ADR<=O'137' .AND. ADR>=O'130') .OR. (ADR<=O'57' .AND. ADR>=O'30'), THEN                                  *
2148 * 11        RETURN ..NOT ACCESSIBLE, THUS NOOP.                                                                      *
2149 * 12    ELSE                                                                                                         *
2150   33 * 13        CALL GET_AREG (U(8), 32_REG(1))                                                                      *
2151 * 14        IF U(AF4)<=O'17' .AND. U(AF4)>=O'11' "INDEX REGISTER REFERENCE", THEN                                    *
2152 * 15            CMR(ADR)(15) := 32_REG(1)(15) ..UPDATE ONLY LOW 16 BITS.                                             *
2153 * 16        ELSE                                                                                                     *
2154 * 17            IF ADR<=O'77' .AND. ADR>=O'70' "ACTIVE STATUS REGISTER", THEN                                        *
2155 * 18                CMR(70)(15) := 32_REG(1)(15) ..CAN ONLY CHANGE LOW 16 BITS.                                      *
2156 * 19            ELSE                                                                                                 *
2157 * 20                IF ADR>=60 .AND. ADR<=( ' "BREAKPOINT REGISTER", THEN                                            *
2158 * 21                    CMR(60) := 32_REG( ) ..ALL 6X REFER TO THE BREAKPOINT REGISTER.                              *
2159 * 22                ELSE ..ALL OTHER REFERENCES.                                                                     *
2160 * 23                    CMR(ADR) := 32_REG(1)                                                                        *
2161 * 24                ENDIF                                                                                            *
2162 * 25            ENDIF                                                                                                *
2163 * 26        ENDIF                                                                                                    *
2164 * 27    ENDIF                                                                                                        *
2165 * 28    RETURN                                                                                                       *
     *                                                                                                                    *
     ********************************************************************************************************************
```

_HLC (& _HRZ & _HRS) SHIFT LEFT CIRCULARLY (RIGHT LOGICAL &

```
REF
PAGE ********************************************************************************************************************
     *                                                                                                                  *
2167 *  1    ..RIGHT ARITHMETIC), FMT IV B, F=62, F=64, F=66                                                           *
2168   24 *  2    CALL GET_SHIFT_AMOUNT (U(M), SHIFT_COUNT) ..ACQUIRE 6-BIT SHIFT COUNT.                                *
2170   33 *  3    CALL GET_AREG (U(A), 32_REG(1))                                                                       *
2171 *  4    DO CASE U(F) ..SHIFT ACCORDING TO OPERATION CODE.                                                         *
2172 *  5        \62\ SHIFT 32_REG(1) LEFT CIRCULAR BY SHIFT_COUNT                                                     *
2173 *  6        \64\ SHIFT 32_REG(1) RIGHT LOGICAL BY SHIFT_COUNT                                                     *
2174 *  7        \66\ SHIFT 32_REG(1) RIGHT ARITHMETIC BY SHIFT_COUNT.                                                 *
2175 *  8    ENDDO ..END CASE.                                                                                         *
2176   36 *  9    CALL PUT_AREG (U(A), 32_REG(1)) ..RESTORE SHIFTED VALUE.                                              *
2177 * 10    CALL HALFWORD_TOGGLE                                                                                      *
2178 * 11    RETURN                                                                                                    *
     *                                                                                                                  *
     ********************************************************************************************************************
```

```
     _HDLC (& _HDRZ & _HDRS) ..SHIFT LEFT DOUBLE CIRCULAR &

     REF
     PAGE ********************************************************************************************************************************
          *
2180      *  1    ..RIGHTLOGICAL & RIGHT ARITHMETIC, FMT IV B, F=63, F=65, F=67                                                          *
2181   24 *  2    CALL GET_SHIFT_AMOUNT (U(M), SHIFT_COUNT) ..ACQUIRE 6-BIT SHIFT COUNT.                                                 *
2182   33 *  3    CALL GET_AREG (U(A)+1, 64_REG(1)(63,32)) ..MOST SIGNIFICANT BITS.                                                      *
2183   33 *  4    CALL GET_AREG (U(A), 64_REG(1)(31)) ..LEAST SIGNIFICANT BITS.                                                          *
2184      *  5    DO CASE U(F) ..SHIFT ACCORDING TO OPERATION CODE.                                                                      *
2185      *  6        \63\ SHIFT 64_REG(1) LEFT CIRCULAR BY SHIFT_COUNT                                                                  *
2186      *  7        \65\ SHIFT 64_REG(1) RIGHT LOGICAL BY SHIFT_COUNT                                                                  *
2187      *  8        \67\ SHIFT64_REG(1) RIGHT ARITHMETIC BY SHIFT_COUNT                                                                *
2188      *  9    ESAC ..END CASE.                                                                                                       *
2189   36 * 10    CALL PUT_AREG (U(A)+1, 64_REG(1)(63,32)) ..MOST SIGNIFICANT BITS.                                                      *
2190   36 * 11    CALL PUT_AREG (U(A), 64_REG(1)(31)) ..LEAST SIGNIFICANT BITS.                                                          *
2191      * 12    CALL HALFWORD_TOGGLE                                                                                                   *
2192      * 13    RETURN                                                                                                                 *
          *                                                                                                                              *
          ********************************************************************************************************************************
```

```
     _HSF ..SCALE FACTOR, FMT IV A, F=70 0

     REF
     PAGE ********************************************************************************************************************
          *                                                                                                                    *
2194   33 *  1    CALL GET_AREG (U(A), 32_REG(1))                                                                             *
2195      *  2    IF 32_REG(1)(31,1), THEN                                                                                    *
2196      *  3        SHIFT_COUNT := (NUMBER OF LEFT JUSTIFIED ONE BITS IN 32_REG(1)) - 1                                     *
2197      *  4    ELSE                                                                                                        *
2198      *  5        SHIFT_COUNT := (NUMBER OF LEFT JUSTIFIED ZERO BITS IN 32_REG(1)) - 1                                    *
2199      *  6    ENDIF                                                                                                       *
2200      *  7    ..NOTE THAT ALL ONES OR ZEROS MEANS THAT SHIFT_COUNT = 0'37'.                                               *
2201      *  8    IF SHIFT_COUNT<0'37', THEN                                                                                  *
2202      *  9        SHIFT LEFT CIRCULARLY 32_REG(1) BY SHIFT_COUNT                                                          *
2203   36 * 10        CALL PUT_AREG (U(A), 32_REG(1)) ..RESTORE SHIFTED VALUE.                                                *
2204      * 11    ENDIF                                                                                                       *
2205      * 12    IF U(A)<>U(B), THEN                                                                                         *
2206   36 * 13        CALL PUT_AREG (U(B), SHIFT_COUNT)                                                                       *
2207      * 14    ENDIF                                                                                                       *
2208      * 15    CALL HALFWORD_TOGGLE                                                                                        *
2209      * 16    RETURN                                                                                                      *
          *                                                                                                                    *
          ********************************************************************************************************************
```

```
_HDSF ..DOUBLE SCALE FACTOR, FMT 1W A, F=70 1

REF
PAGE ********************************************************************************************************************************
     *
2211   33 *  1    CALL GET_AREG(U(A)+1, 64_REG(I)(63:32))
2212   33 *  2    CALL GET_AREG(U(A), 64_REG(I)(31))
2213      *  3    IF 64_REG(I)(63:1), THEN
2214      *  4        SHIFT_COUNT := (# OF LEFT JUSTIFIED ONES) - 1
2215      *  5    ELSE
2216      *  6        SHIFT_COUNT := (# OF LEFT JUSTIFIED ZEROS) - 1
2217      *  7    ENDIF
2218      *  8    IF SHIFT_COUNT < D'77', THEN
2219      *  9        SHIFT LEFT CIRCULARLY (64_REG(I), SHIFT_COUNT)
2220   36 * 10        CALL PUT_AREG (U(A)+1, 64_REG(I)(63:32))
2221   36 * 11        CALL PUT_AREG (U(A), 64_REG(I)(31:0))
2222      * 12    ENDIF
2223      * 13    IF U(A)<>U(B) .AND. (U(A)+1)<>U(B), THEN
2224   36 * 14        CALL PUT_AREG(U(B),SHIFT_COUNT)
2225      * 15    ENDIF
2226      * 16    CALL HALFWORD_TOGGLE
2227      * 17    RETURN
     *
     ********************************************************************************************************************************
```

```
      _HCP ..COMPLEMENT A, FMT IV A, F=70 2

      REF
      PAGE ********************************************************************************************************************
           *                                                                                                                    *
2229    33 *  1    CALL GET_AREG(U(A), 32_REG(1))                                                                              *
2230       *  2    32_REG(1) := .NOT. 32_REG(1)                                                                                *
2231    36 *  3    CALL PUT_AREG(U(A), 32_REG(1))                                                                              *
2232       *  4    CALL HALFWORD_TOGGLE                                                                                        *
2233       *  5    RETURN                                                                                                      *
           *                                                                                                                    *
           ********************************************************************************************************************
```

_HDCP .. DOUBLE COMPLEMENT A, FMT IV A, F=70 3

```
REF
PAGE ***************************************************************************************************
     *                                                                                                   *
2235   33 *  1    CALL GET_AREG(U(A)+1, 32_REG(2))                                                      *
2236      *  2    32_REG(2) := .NOT. 32_REG(2)                                                          *
2237   36 *  3    CALL PUT_AREG(U(A)+1, 32_REG(2))                                                      *
2238  125 *  4    CALL _HCP                                                                             *
2239      *  5    RETURN                                                                                *
     *                                                                                                   *
     ***************************************************************************************************
```

_HOR(& _HA,_HAN,_HICP,_HAND), FMT IV A, F=71 0 THRU 71 3 & F=71 5

```
REF
PAGE ********************************************************************************************************************
     *                                                                                                                    *
2241   33 *  1    CALL GET_AREG(U(A), 32_REG(1))                                                                         *
2242   33 *  2    CALL GET_AREG(U(B), 32_REG(2))                                                                         *
2243      *  3    DO _CASE U(F4)                                                                                         *
2244      *  4        \0\ LOGICAL SUM                                                                                    *
2245      *  5        32_REG(1) := 32_REG(1) .V. 32_REG(2)                                                               *
2246      *  6        \1\ SUM                                                                                            *
2247      *  7        32_REG(1) := 32_REG(1) + 32_REG(2) ..ONES COMPLEMENT ADDITION.                                     *
2248      *  8        UPDATE FIXED POINT OVERFLOW INDICATOR(ASR(3,1))                                                    *
2249      *  9        \2\ DIFFERENCE                                                                                     *
2250      * 10        32_REG(1) := 32_REG(1) - 32_REG(2) ..ONES COMPLEMENT SUBTRACTION.                                  *
2251      * 11        UPDATE FIXED POINT OVERFLOW INDICATOR(ASR(3,1))                                                    *
2252      * 12        \3\ LOGICAL DIFFERENCE                                                                             *
2253      * 13        32_REG(1) := 32_REG(1) .X. 32_REG(2)                                                               *
2254      * 14        \5\ AND                                                                                            *
2255      * 15        32_REG(1) := 32_REG(1) .A. 32_REG(2)                                                               *
2256      * 16    ENDDO .. _ENDCASE                                                                                      *
2257   36 * 17    CALL PUT_AREG(U(A), 32_REG(1))                                                                         *
2258      * 18    CALL HALFWORD_TOGGLE                                                                                   *
2259      * 19    RETURN                                                                                                 *
     *                                                                                                                    *
     ********************************************************************************************************************
```

```
_HM ..MULTIPLY REGISTER, FMT IV A, F=74 0

REF
PAGE ********************************************************************************************************************************
     *                                                                                                                              *
2261 *  1    .. A(U(A)) X A(U(B)) -> A(U(A)+1); A(U(A))                                                                           *
2262   33 *  2    CALL GET_AREG (U(A), 32_REG(2)) ..GET MULTIPLICAND.                                                               *
2263   33 *  3    CALL GET_AREG (U(B), 32_REG(1)) ..GET MULTIPLIER.                                                                 *
2264 *  4    SIGN_INDICATOR := 32_REG(1)(31,1) .X. 32_REG(2)(31,1) ..REMEMBER SIGN FOR RESULT.                                     *
2266   98 *  5    CALL DO_MULTIPLY ..DO ACTUAL MULTIPLICATION                                                                       *
2267 *  6    IF SIGN_INDICATOR "NEGATIVE", THEN ..NEGATE ANSWER.                                                                   *
2268 *  7        REPEAT_ACCUMULATOR := .NOT. REPEAT_ACCUMULATOR                                                                    *
2269 *  8        32_REG(1) := .NOT. 32_REG(1)                                                                                      *
2270 *  9    ENDIF                                                                                                                 *
2271   36 * 10    CALL PUT_AREG (U(A)+1, REPEAT_ACCUMULATOR) ..STORE MOST SIGNIFICANT BITS                                          *
2272   36 * 11    CALL PUT_AREG (U(A), 32_REG(1)) ..STORE LEAST SIGNIFICANT BITS                                                    *
2273 * 12    CALL HALFWORD_TOGGLE                                                                                                  *
2274 * 13    RETURN                                                                                                                *
     *                                                                                                                              *
     ********************************************************************************************************************************
```

```
      _MD ..DIVIDE REGISTER FMT IV A, F=74 1

      REF
      PAGE ***************************************************************************************************
           *                                                                                                   *
2276       *  1    ..(A(U(A+1)), A(U(A))) / A(U(B)) -> A(U(A)); REMAINDER -> A(U(A+1))                         *
2277    33 *  2    CALL GET_AREG(U(B), 32_REG(2)) ..GET DIVISOR.                                               *
2278    33 *  3    CALL GET_AREG(U(A), 64_REG(1)(31,32)) ..GET DIVIDEND.                                       *
2279    33 *  4    CALL GET_AREG(U(A)+1, 64_REG(1)(63,32))                                                     *
2280    97 *  5    CALL DO_DIVIDE (64_REG(1), 32_REG(2)) ..DO ACTUAL DIVISION.                                 *
2281    36 *  6    CALL PUT_AREG(U(A)+1, 64_REG(1)(63,32)) ..STORE REMAINDER.                                  *
2282    36 *  7    CALL PUT_AREG(U(A), 64_REG(1)(31,32)) ..STORE QUOTIENT.                                     *
2283       *  8    CALL HALFWORD_TOGGLE                                                                        *
2284       *  9    RETURN                                                                                      *
           *                                                                                                   *
           ***************************************************************************************************
```

```
     _HRT ..SQUARE ROOT FMT IV A F=74 2

     REF
     PAGE ********************************************************************************************************
          *                                                                                                      *
2286      *  1    .. THIS INSTRUCTION USES THE ALGORITHMS ON PAGE 445-452 OF THE UYK-7 LEARNER'S GUIDE          *
2288      *  2    ..(JANUARY 1973). UPON COMPLETION 32_REG(1) EQUALS THE SQUARE ROOT AND 64_REG(1)(63,32)       *
2290   33 *  3    CALL GET_AREG(U(A)+1, 64_REG(1)(31,32)) ..GET HIGH ORDER BITS OF RADICAND.                     *
2292      *  4    IF 64_REG(1)(31,2) "UPPER 2 BITS OF RADICAND", THEN                                            *
2293      *  5        ASR(3,1) := 1 ..INDICATE OVERFLOW.                                                         *
2294      *  6    ELSE                                                                                           *
2295      *  7        ASR(3,1) := 0 ..CLEAR OVERFLOW.                                                            *
2296      *  8    ENDIF                                                                                          *
2297      *  9    64_REG(1)(63,32) := 0 ..INITIALIZE THE CURRENT RESIDUE.                                        *
2298      * 10    32_REG(1) := 0 ..INITIALIZE THE ROOT.                                                          *
2299  131 * 11    CALL DO_SQRT(64_REG(1), 32_REG(1)) ..WORK ON HIGH ORDER BITS OF THE RADICAND.                  *
2301      * 12    ..NOTE: DO_SQRT USES 32_REG(2) FOR SCRATCH.                                                    *
2302   33 * 13    CALL GET_AREG(U(A), 64_REG(1)(31,32)) ..GET LEAST SIGNIFICANT BITS RADICAND.                   *
2304  131 * 14    CALL DO_SQRT(64_REG(1), 32_REG(1)) ..WORK ON LOW BITS OF RADICAND.                             *
2305      * 15    32_REG(1) = 32_REG(1) .RL.1 ..COMPENSATE FOR EXTRA SHIFT IN DO_SQRT.                           *
2306   36 * 16    CALL PUT_AREG(U(B), 32_REG(1)) ..STORE ROOT.                                                   *
2307   36 * 17    CALL PUT_AREG(U(B)+1,64_REG(1)(63,32)) ..STORE RESIDUE.                                        *
2308      * 18    CALL HALFWORD_TOGGLE                                                                           *
2309      * 19    RETURN                                                                                         *
          *                                                                                                      *
          ********************************************************************************************************
```

```
      DO_SQRT (REF 64_REG, REF 32_REG)

      REF
      PAGE ********************************************************************************************************************
           *                                                                                                                    *
2311       *  1    COUNT := 16                                                                                                 *
2312       *  2    DO WHILE COUNT > 0                                                                                          *
2313       *  3        64_REG := 64_REG .LL. 2 ..THE PARTIAL RADICAND (64_REG(63,32)) IS FORMED BY SHIFTING THE                *
2315       *  4        ..CURRENT RESIDUE (64_REG(63,32)) LEFT TWO BITS AND SUBSTITUTING TWO BITS FROM THE ORIGINAL             *
2317       *  5        ..RADICAND (64_REG(31,32)). THE PARTIAL RADICAND (64_REG(63,32)) MAY REQUIRE MORE THAN                  *
2319       *  6        ..32 BITS TO EXPRESS.                                                                                   *
2320       *  7        32_REG := 32_REG .LL. 1 ..SHIFT PARTIAL ROOT LEFT 1.                                                    *
2321       *  8        32_REG(2) := 32_REG + 1 ..32_REG(2) IS THE 32 BIT TRIAL ROOT EXTRACTOR.                                 *
2322       *  9        IF 32_REG >= 32_REG(2), THEN ..UNSIGNED 34 BIT COMPARE (TO BE SAFE).                                    *
2323       * 10            32_REG := 32_REG - 32_REG(2) ..32_REG NOW HAS THE NEW CURRENT RESIDUE.                              *
2324       * 11            32_REG(1,1) := 1 ..SET BIT IN ROOT.                                                                 *
2325       * 12        ENDIF                                                                                                   *
2326       * 13        COUNT := COUNT-1                                                                                        *
2327       * 14    ENDDO                                                                                                       *
2328       * 15    RETURN                                                                                                      *
           *                                                                                                                    *
           ********************************************************************************************************************
```

```
_HLB ..LOAD B(A) WITH B(B) FMT IV A, F=74 3

REF
PAGE ********************************************************************************************************************
     *                                                                                                                  *
2330 *  1    ..B(U(B)) -> B(U(A))                                                                                       *
2331   34 *  2    CALL GET_BREG(U(B), 32_REG(1))                                                                        *
2332   34 *  3    CALL GET_BREG(U(A), 32_REG(2))                                                                        *
2333 *  4    32_REG(2)(15) := 32_REG(1)(15)                                                                             *
2334   37 *  5    CALL PUT_BREG(U(A), 32_REG(2))                                                                        *
2335 *  6    CALL HALFWORD_TOGGLE                                                                                       *
2336 *  7    RETURN                                                                                                     *
     *                                                                                                                  *
     ********************************************************************************************************************
```

```
     _HC ..COMPARE REGISTER, FMT IV A, F=74 4

     REF
     PAGE ********************************************************************************************************************
          *                                                                                                                    *
2338      *  1    .. A(U(A)) IS COMPARED TO A(U(B)) ; SET ASR(2,2) "CO" ACCORDINGLY.                                          *
2339   33 *  2    CALL GET_AREG(U(A), 32_REG(1))                                                                              *
2340   33 *  3    CALL GET_AREG(U(B), 32_REG(2))                                                                              *
2341   29 *  4    CALL SET_CO1(32_REG(1), 32_REG(2))                                                                          *
2342      *  5    CALL HALFWORD_TOGGLE                                                                                        *
2343      *  6    RETURN                                                                                                      *
          *                                                                                                                    *
          ********************************************************************************************************************
```

```
     _HCL ..COMPARE LIMITS, REGISTER FMT IV A, F=74 5

     REF
     PAGE ********************************************************************************************************************
          *                                                                                                                    *
2345      *  1    .. A(U(A)+1) > A(U(B)) >= A(U(A)) -> SET ASP(C,1) "OUTSIDE/WITHIN LIMITS.                                    *
2347   33 *  2    CALL GET_AREG(U(A), 32_REG(2))                                                                               *
2348   33 *  3    CALL GET_AREG(U(A)+1, 32_REG(3))                                                                             *
2349   33 *  4    CALL GET_AREG(U(B), 32_REG(1))                                                                               *
2350   30 *  5    CALL SET_CD2(32_REG(1), 32_REG(2), 32_REG(3))                                                                *
2351      *  6    CALL HALFWORD_TOGGLE                                                                                         *
2352      *  7    RETURN                                                                                                       *
          *                                                                                                                    *
          ********************************************************************************************************************
```

```
     _HCM ..COMPARE MASKED, REGISTER FMT IV A, F=74 6

     REF
     PAGE ********************************************************************************************************************
          *                                                                                                                    *
2354      *  1    .. A(U(A)+1) .A. A(U(A)) IS COMPARED TO A(U(B)) ; SET ASR(Z,2) "CD" ACCORDINGLY.                            *
2356   33 *  2    CALL GET_AREG(U(A), 32_REG(1))                                                                              *
2357   33 *  3    CALL GET_AREG(U(A)+1, 32_REG(2))                                                                            *
2358      *  4    32_REG(1) := 32_REG(1) .AND. 32_REG(2)                                                                      *
2359   33 *  5    CALL GET_AREG(U(B), 32_REG(2))                                                                              *
2360   29 *  6    CALL SET_CD1(32_REG(1), 32_REG(2))                                                                          *
2361      *  7    CALL HALFWORD_TOGGLE                                                                                        *
2362      *  8    RETURN                                                                                                      *
          *                                                                                                                    *
          ********************************************************************************************************************
```

```
     _HCB ..COMPARE B(U(B)) WITH B(U(A)) FMT IV A F=74 7

REF
PAGE ********************************************************************************************************************************
     *                                                                                                                              *
2364 *  1    .. B(U(B))(15,16) IS COMPARED TO B(U(A))(15,16) ; SET ASR(2,2) "CD" ACCORDINGLY                                        *
2366   34 *  2    CALL GET_BREG(U(B), 32_REG(1))                                                                                    *
2367   34 *  3    CALL GET_BREG(U(A), 32_REG(2))                                                                                    *
2368 *  4    32_REG(1) := 32_REG(1)(15)                                                                                             *
2369 *  5    32_REG(2) := 32_REG(2)(15)                                                                                             *
2370   29 *  6    CALL SET_CD1(32_REG(1), 32_REG(2))                                                                                *
2371 *  7    CALL HALFWORD_TOGGLE                                                                                                   *
2372 *  8    RETURN                                                                                                                 *
     *                                                                                                                              *
     ********************************************************************************************************************************
```

```
     _HSIM ..STORE ICC MONITOR CLOCK IN A PPT IV A P=77 0

     REF
     PAGE ********************************************************************************************************************
          *                                                                                                                    *
2374      *  1    IF ASP(19,4)=0 "TASK MODE", THEN ..PRIVILEGED INSTRUCTION.                                                   *
2375   11 *  2        CALL GENERATE_SYNCHRONOUS_INTERRUPT ("P-"1, PRIVILEGED INSTRUCTION VIOLATION)                            *
2377      *  3        ..ABORT THE INSTRUCTION.                                                                                 *
2378      *  4    ENDIF                                                                                                        *
2379   22 *  5    CALL CP/ICC_CLOCK_COMMUNICATIONS                                                                             *
2380      *  6    GET G_BUS INTO 32_REG(2)                                                                                     *
2381   36 *  7    CALL PUT_AREG(U(8), 32_REG(2))                                                                               *
2382      *  8    CALL HALFWORD_TOGGLE                                                                                         *
2383      *  9    RETURN                                                                                                       *
          *                                                                                                                    *
          ********************************************************************************************************************
```

```
      _HSTC ..STORE REAL-TIME CLOCK IN A FMT IV A F=77 I

      REF
      PAGE ***************************************************************************************************************
           *                                                                                                             *
2385    22 *  1    CALL CP/IOC_CLOCK_COMMUNICATIONS                                                                      *
2386       *  2    GET O_BUS INTO 32_REG(2)                                                                              *
2387    36 *  3    CALL PUT_AREG(U(8), 32_REG(2))                                                                        *
2388       *  4    CALL HALFWORD_TOGGLE                                                                                  *
2389       *  5    RETURN                                                                                                *
           *                                                                                                             *
           ***************************************************************************************************************
```

_HPI(E_HAI) ..PREVENT(ALLOW) CLASS III INT. FMT IV A, F=77 & E F=77 5

```
REF
PAGE ********************************************************************************************************************
     *                                                                                                                  *
2391 *  1    IF ASR(19,4)=0 "TASK MODE", THEN ..PRIVILEGED INSTRUCTION.                                                  *
2392 * 11 *  2        CALL GENERATE_SYNCHRONOUS_INTERRUPT("P-"1, PRIVILEGED INSTRUCTION VIOLATION)                      *
2394 *  3        ..ABORT THE INSTRUCTION.                                                                                *
2395 *  4    ENDIF                                                                                                       *
2396 *  5    IF L(F4) = 4 "HPI", T=EN                                                                                    *
2397 *  6        ASR(12,1) := 1 ..LOCKOUT CLASS III INTERRUPTS.                                                          *
2398 *  7    ELSE ..="AI".                                                                                               *
2399 *  8        ASR(12.1) := 0 ..ENABLE CLASS III INTERRUPTS.                                                           *
2400 *  9    ENDIF                                                                                                       *
2401 * 10    CALL HALFWORD_TOGGLE                                                                                        *
2402 * 11    RETURN                                                                                                      *
     *                                                                                                                  *
     ********************************************************************************************************************
```

```
     _H776 ..HALT AND HALT AND WAIT FOR INTERRUPT FMT IV A F=77 6

     REF
     PAGE ********************************************************************************************************************
          *                                                                                                                    *
2404      *  1    IF ASP(19,4)=0 "TASK MODE", THEN ..PRIVILEGED INSTRUCTION.                                                   *
2405   11 *  2        CALL GENERATE_SYNCHRONOUS_INTERRUPT("P-"1, PRIVILEGED INSTRUCTION VIOLATION)                           *
2407      *  3        ..ABORT THE INSTRUCTION.                                                                              *
2408      *  4    ENDIF                                                                                                     *
2409      *  5    IF .NOT. U(I), THEN ..HALT INSTRUCTION.                                                                   *
2410      *  6        STOP PROCESSING INSTRUCTIONS WITH STOP & INDICATOR SET                                                *
2411      *  7    ELSE ..HALT AND WAIT FOR INTERRUPT.                                                                       *
2412      *  8        STOP PROCESSING INSTRUCTIONS                                                                          *
2413      *  9        WAIT FOR AN ASYNCHRONOUS OR CP MONITOR CLOCK INTERRUPT                                                *
2414      * 10        UPON INTERRUPT PROCESS IT AND CONTINUE WITH NEXT INSTRUCTION                                          *
2415      * 11    ENDIF                                                                                                     *
2416      * 12    CALL HALFWORD_TOGGLE                                                                                      *
2417      * 13    RETURN                                                                                                    *
          *                                                                                                                    *
          ********************************************************************************************************************
```

AN/UYK-7 CP EMULATION GLOSSARY

AN/UYK-7 CP EMULATION STATUS/CONTROL INDICATORS

```
2420        1       TO SUPPORT REENTRANCY ALL GLOBAL CONTROL VARIABLES ARE CONSIDERED
2421        2    TO BE ACCESSIBLE BY ALL PROCEDURES BUT CONTAINED WITHIN A WORK SPACE
2422        3    FOR A PARTICULAR EMULATOR.
2423        4
2424        5    CHARACTER_ADDRESSING_OVERRIDE - WHEN SET, CHARACTER ADDRESSING IS
2425        6                                    IN EFFECT.  USED TO ACQUIRE CORRECT
2426        7                                    CHARACTER SPECIFICATION VALUES (I.E.
2427        8                                    _C, _P, AND _MASK) WHEN REPEATING
2428        9                                    INSTRUCTIONS USING CHARACTER ADDRESSING.
2429       10    CP_MONITOR_CLOCK_INTERRUPT -  WHEN SET, THE CP MONITOR CLOCK HAS GONE
2430       11                                  NEGATIVE AND THE ASSOCIATED CLASS II
2431       12                                  INTERRUPT IS HELD PENDING UNTIL PROCESSED.
2432       13    EXECUTE_REMOTE_IN_PROGRESS -  WHEN SET, AN INSTRUCTION IS BEING EXECUTED
2433       14                                  VIA THE XR (F=02 2) OR XRL (F=02 3)
2434       15                                  INSTRUCTION.  USED TO SUPPESS CHANGING OF
2435       16                                  THE ASR UPPER/LOWER HALF-WORD BIT WHEN A
2436       17                                  HALF-WORD INSTRUCTION IS EXECUTED VIA THE
2437       18                                  XR OR XRL INSTRUCTION.
2438       19    INSTRUCTION_FORMAT_INDICATOR -  THIS INDICATOR CONTAINS THE FORMAT
2439       20                                    (I.E. I, II, III, IV) OF THE INSTRUCTION
2440       21                                    THAT IS CURRENTLY EXECUTING.
2441       22    INTERPROCESSOR_INTERRUPT    -  WHEN SET, AN INTERPROCESSOR INTERRUPT IS
2442       23                                  PENDING.  THIS DATA STRUCTURE MUST RECEIVE
2443       24                                  THE CP ID OF THE PROCESSOR CAUSING THE
2444       25                                  INTERRUPT.
2445       26    INTERRUPT_SCAN_INHIBIT      -   WHEN SET (I.E. BETWEEN EXECUTION OF
2446       27                                  TWO HALF-WORD INSTRUCTIONS AND DURING
2447       28                                  REPEATS OF CMR REFERENCE INSTRUCTIONS
2448       29                                  (F=54-57)), THE ASYNCHRONOUS INTERRUPT
2449       30                                  SCAN IS NOT PERFORMED.  FURTHERMORE, THE
2450       31                                  MAIN LOOP LOGIC SUPPRESSES INTERRUPT
2451       32                                  SCANNING WHEN IN INTERRUPT MODE.
```

STATUS/CONTROL INDICATORS PAGE 2

```
2453    +  1    MEMORY_STORE_INDICATOR      -   THIS INDICATOR IS SET WHEN AN INSTRUCTION
2454    +  2                                 STORES A QUANTITY INTO MEMORY. THIS FACT ,
2455    +  3                                 IS REQUIRED FOR REPEAT TERMINATION LOGIC.
2456    +  4    PSEUDO_BREAKPOINTS          -   WHEN SET, ALL BREAKPOINT REGISTERS WILL
2457    +  5                                 BE ACTIVE (TOTAL OF 8), ELSE ONLY THE
2458    +  6                                 EMULATED HARDWARE REGISTER WILL BE ACTIVE.
2459    +  7                                 NOTE, THE UYK-7 EMULATOR HAS EIGHT (8)
2460    +  8                                 BREAKPOINT REGISTERS.  ONE REGISTER
2461    +  9                                 CORRESPONDS TO THE ACTUAL HARDWARE
2462    + 10                                 BREAKPOINT REGISTER OF A REAL UYK-7 MACHINE.
2463    + 11                                 THE OTHER SEVEN (7) BREAKPOINT REGISTERS
2464    + 12                                 ARE CALLED PSEUDO BREAKPOINT REGISTERS AND
2465    + 13                                 ARE CONSIDERED EXTENSIONS TO THE UYK-7
2466    + 14                                 ARCHITECTURE.  PSEUDO BREAKPOINTS ARE NOT
2467    + 15                                 ACCESSIBLE TO UYK-7 PROGRAMS.
2468    + 16    REPEAT_IN_PROGRESS          -   WHEN SET, INSTRUCTION IS BEING REPEATED.
2469    + 17    REPEAT_PENDING              -   SET BY EXECUTION OF A REPEAT INSTRUCTION.
2470    + 18                                 REPEAT IS NOT IN PROGRESS UNTIL THE REPEATED
2471    + 19                                 INSTRUCTION HAS BEEN FETCHED FROM MEMORY.
2472    + 20    SPR_CHECKS                  -   WHEN SET, SOFTWARE PROTECTION REGISTER
2473    + 21                                 CHECKS ARE ENABLED. SPR CHECKS ARE
2474    + 22                                 DISABLED WHEN CLASS II INTERRUPTS ARE
2475    + 23                                 LOCKED OUT, INTERRUPT BASE REGISTERS ARE
2476    + 24                                 SELECTED OR MEMORY LOCKOUT INHIBIT IS SET.
2477    + 25    SPR_PRIVILEGED_INSTRUCTION -   WHEN SET, ONE OF THE CONDITIONS CAUSING
2478    + 26                                 USE OF INDIRECT ADDRESSING WITH SPR(16,1)
2479    + 27                                 TO BE A PRIVILEGED OPERATION IS IN EFFECT.
2480    + 28                                 THESE CONDITIONS INCLUDE:
2481    + 29                                    (1) CERTAIN DOUBLE-WORD INSTRUCTIONS
2482    + 30                                          (ALL EXCEPT OS 4).
2483    + 31                                    (2) ALL REPEATED INSTRUCTIONS.
2484    + 32                                    (3) U(F) = 25 OR U(F) = 11.
2485    + 33                                    (4) ALL FORMAT III INSTRUCTIONS.
```

AN/UYK-7 CP ARCHITECTURE DESCRIPTORS

```
2487    1   BREAKPOINT_REGISTER     -  CMR ADDRESS OF UYK-7 HARDWARE BREAKPOINT
2488    2                           REGISTER (60).
2489    3   _CMR                    -  (CONTROL MEMORY REGISTERS) SECTION OF CP
2490    4                           WHERE THE ADDRESSABLE REGISTERS ARE LOCATED.
2491    5                           INCLUDED ARE TASK AND INTERRUPT
2492    6                           ACCUMULATORS, INDEX(8), AND BASE(S)
2493    7                           REGISTERS.  SPR'S AND SIR'S, THE ASP, CP
2494    8                           MONITOR CLOCK AND BREAKPOINT REGISTER(A)
2495    9                           AND ICW AND OSW FOR EACH INTERRUPT LEVEL.
2496   10                           CMR LOCATION 10 IS UNASSIGNED BIT
2497   11                           ADDRESSABLE BY PROGRAMS.
2498   12   _MAIN                  -  MAIN MEMORY SUPPLYING 32-BIT WORD
2499   13                           STORAGE.
2500   14   MEMORY_LIMIT           -  AMOUNT OF 32-BIT MAIN MEMORY (UP TO
2501   15                           256 K) AVAILABLE FOR USE BY CP
2502   16   _NDRO                  -  512 WORD NON-DESTRUCTIVE READ OUT
2503   17                           MEMORY.  USUALLY CONTAINS DIAGNOSTIC,
2504   18                           BOOTSTRAP, RECOVERY AND INTERRUPT SWITCHER
2505   19                           PROGRAMS.
```

AN/UYK-7 CP MAINTENANCE PANEL SUPPORT

```
2507    +  1    BOOTSTRAP_SWITCH          -   A 3-POSITION SWITCH (0,1,2) WHICH                +
2508    +  2                               SELECTS AN NDRC STARTING ADDRESS.                   +
2509    +  3    _STEP                     -   WHEN SET, THE EMULATOR SUSPENDS OPERATION        +
2510    +  4                               AT THE BEGINNING OF THE MAIN LOOP UNTIL             +
2511    +  5                               RESTARTED. THIS CORRESPONDS TO THE MODE             +
2512    +  6                               SWITCH (I.E. SET -> INSTRUCTION MODE                +
2513    +  7                               AND CLEAR -> RUN MODE).                             +
2514    +  8    STOP_SWITCH               -   SERIES OF THREE SWITCHES (NUMBERED              +
2515    +  9                               5,6,7) FOUND ON THE MAINTENANCE PANEL               +
2516    + 10                               WHICH ALLOW STOPPING UNDER PROGRAM CONTROL.         +
2517    + 11    JUMP_SWITCH               -   SERIES OF THREE SWITCHES (NUMBERED              +
2518    + 12                               1,2,3) FOUND ON THE MAINTENANCE PANEL               +
2519    + 13                               WHICH ALLOW BRANCHING UNDER PROGRAM CONTROL.        +
```

AN/UYK-7 CP EMULATION INTERNAL REGISTER SUPPORT

```
2521  +  1   _C                    -   THE C DESIGNATOR FOR CHARACTER ADDRESSING
2522  +  2                          C FIELD OF LAST EXECUTED IAW.
2523  +  3   M                     -   THE MEMORY REFERENCE REGISTER.  A 32 BIT
2524  +  4                          REGISTER CONTAINING THE ADDRESS OF THE LAST
2525  +  5                          MEMORY LOCATION REFERENCED.
2526  +  6   _MASK                 -   THE CHARACTER MASK NEEDED TO ACQUIRE
2527  +  7                          6 BITS FOR CHARACTER ADDRESSING.
2528  +  8   P                     -   THE PROGRAM COUNTER REGISTER.  A 20-BIT
2529  +  9                          REGISTER WITH LOW ORDER 16 BITS (P(0))
2530  + 10                          FORMING A DISPLACEMENT THAT IS ADDED TO THE
2531  + 11                          BASE REGISTER INDICATED BY BITS 19 THROUGH
2532  + 12                          17.
2533  + 13                          BIT I6 IS UNUSED.
2534  + 14   _P                    -   LEAST SIGNIFIENT BIT ADDRESS FOR
2535  + 15                          CHARACTER ADDRESSING.
2536  + 16   REPEAT_AB              -   CONTAINS THE A & B FIELDS OF THE LAST
2537  + 17                          REPEAT INSTRUCTION AS USED IN REPEAT
2538  + 18                          TERMINATION LOGIC.
2539  + 19   REPEAT_ACCUMULATOR     -   A 32-BIT WORKING REGISTER WHICH HOLDS
2540  + 20                          A COPY OF THE APPROPRIATE ACCUMULATOR TO
2541  + 21                          TESTED BY REPEAT TERMINATION LOGIC. IT IS
2542  + 22                          THE RESPONSIBILITY OF EACH INSTRUCTION
2543  + 23                          (WHERE APPLICABLE) TO ENSURE THE PROPER
2544  + 24                          CONTENTS OF THIS REGISTER.
2545  + 25   REPEAT_SY             -   CONTAINS THE SY MODIFIER OF THE LAST
2546  + 26                          REPEAT INSTRUCTION AS USED IN REPEAT
2547  + 27                          TERMINATION LOGIC.
2548  + 28   U                     -   THE INSTRUCTION REGISTER.  A 32-BIT
2549  + 29                          REGISTER WHICH RECEIVES THE INSTRUCTION WORD
2550  + 30                          FROM MEMORY AND PROVIDES THE CONTROL
2551  + 31                          NECESSARY TO START EXECUTION OF THE
2552  + 32                          INSTRUCTION.  FIELDS (E.G. F, F2, A, ETC.)
2553  + 33                          ARE AS PER REPERTOIRE CARD.
2554  + 34   UL                    -   LOWER 16 BITS OF U.
2555  + 35   UU                    -   UPPER 16 BITS OF U.
2556  + 36
2557  + 37     NOTE:   32_REG IS A GENERIC TERM FOR A 32-BIT REGISTER. THIS DESIGN
2558  + 38           ASSUMES 4 SUCH WORKING REGISTERS WHICH ARE REFERENCED AS
2559  + 39           32_REG(1),....,32_REG(4).  THESE REGISTERS DO NOT NECESSARILY
2560  + 40           CORRESPOND TO ANY SPECIFIC UYK-7 INTERNAL REGISTERS.
2561  + 41              REFERENCE PARAMETERS ARE INDICATED BY THE 'REF'
2562  + 42           CONSTRUCT IN THE SUBPROCEDURE DECLARATION.  CHANGING A
2563  + 43           REFERENCE PARAMETER MODIFIES THE ARGUMENT IN THE CALLING
2564  + 44           ROUTINE.
```

REFERENCES

REFERENCES

1. CAINE, FARBER, AND GORDON, "PROGRAM DESIGN LANGUAGE REFERENCE GUIDE", JULY 1975.

2. NAVSHIPS 0967-051-6291, AN/UYK-7(V) COMPUTER SET, MARCH 1977.

3. SPERRY UNIVAC, STUDY GUIDE FOR AN/UYK-7 COMPUTER MAINTENANCE COURSE VOLUMES 1, 2 & 3, JANUARY 1977.

4. NAVSHIPS 0967-319-4012, TECHNICAL MANUAL FOR COMPUTER SET AN/UYK-7(V) VOLUMES 1 & 2, JANUARY 1974.

5. SPERRY UNIVAC, STUDY GUIDE FOR AN/UYK-7 COMPUTER, VOLUME 1, MARCH 1973.

INDEX TO DATA ITEMS

INDEX TO DATA ITEMS

INDEX TO DATA ITEMS

INDEX TO DATA ITEMS

INDEX TO DATA ITEMS

INDEX TO DATA ITEMS

INDEX TO DATA ITEMS

PAGE	LINE	TYPE	NAME AND REFERENCES
			109 _LBJ
			8 10
			110 _JBNZ
			6 8
			112 _JL
			4 7
			113 _J530
			2 7
			114 _J531
			3 32
			115 _J532
			24 27
			116 J533
			2 23
		DI	OPERAND_S
			13 JUMP_ADDRESS
			4 7 12 15 16 17
			55 _XR
			3 7
			105 _JEP
			2 16
			106 _DJZ
			2 7
			107 _DJNZ
			2 7
			108 _F51
			2 13
			109 _LBJ
			8 10
			110 _JBNZ
			6 8
			112 _JL
			4 6
			113 _J530
			2 7
			114 _J531
			3 32
			115 _J532
			24 27
			116 J533
			2 23
		DI	OPERAND_DISPLACEMENT
			105 _JEP
			16

INDEX TO DATA ITEMS

INDEX TO DATA ITEMS

PAGE	LINE	TYPE	NAME AND REFERENCES
			49 _SC
			3 4 4 5
			50 _MS
			3 5 5 6
			51 _XOR
			2 3 3 4
			52 _ALP
			1 4 4 6
			53 _LLP
			3 5 5 8 8 12 14
			54 _CNT
			2 6 6 10
			56 _SLP
			1 3 3 4
			57 _SSUM
			1 3 3 9 10
			58 _SDIF
			2 3 3 9 10
			60 _TSF
			4 5 10 11
			67 _FM
			14 15 16 17 21 24 24 26
			76 _LA
			1 2
			77 _LXB
			2 3
			78 _LDIF
			1 3 3 5
			79 _ANA
			2 3 3 5
			80 _AA
			2 3 3 5
			81 _LSUM
			2 3 3 5
			82 _LNA
			1 2 2 3
			83 _LM
			1 2 3 3 5
			85 _AB
			4 6 7
			86 _ANB
			3 4 6 7
			87 _SB
			2 3 3 4
			88 _SA
			2 3
			90 _SNA
			2 3 3

INDEX TO DATA ITEMS

INDEX TO DATA ITEMS

INDEX TO DATA ITEMS

INDEX TO DATA ITEMS

PAGE	LINE	TYPE	NAME AND REFERENCES
			110 _JBNZ 1
			112 _JL 1
			113 _J530 1
			114 _J531 2
			115 _J532 1
			116 J533 1
		DI	STOP_SWITCH 115 _J532 10
			116 J533 11
		DI	S_DESIGNATOR 14 OP_READ 6 11 19 20 22
			15 OP_STORE 6 11 21 22 24
			18 IA_SEQUENCE 2 7 9 11 13 18 20 22 24
			19 MEMORY_READ 4 7
			23 DO_JUMP 3 5
			32 ADD_S 2 7 8
			35 GET_SREG 1 2 5 5 7 9
			38 PUT_SREG 1 2 4 4 6 8
			73 _IO 6 8 12
		DI	S_INDICATOR 20 SPP_CHECK 8 13
		DI	TYPE_ 17 SPP_CHECK 8 18 26 31 38

INDEX TO DATA ITEMS

INDEX TO DATA ITEMS

INDEX TO DATA ITEMS

INDEX TO DATA ITEMS

INDEX TO DATA ITEMS

INDEX TO DATA ITEMS

INDEX TO DATA ITEMS

INDEX TO DATA ITEMS

PAGE	LINE	TYPE	NAME AND REFERENCES
			18 IA_SEQUENCE
			4 36
		DI	_RNLP
			53 _LLP
			6
		DI	_STEP
			3 CP MAIN LOOP
			2
			17 BPR_CHECK
			30
		DI	_WHILE
			54 _CNT
			4
			105 _JEP
			10
		DI	_XRL
			55 _XR
			11 14

INDEX TO FLOW SEGMENTS

INDEX TO FLOW SEGMENTS

INDEX TO FLOW SEGMENTS

INDEX TO FLOW SEGMENTS

PAGE	LINE	TYPE	NAME AND REFERENCES
131		FS	DG_SORT
			130 _HRT
			11 14
40		FS	FLOATING_ADD_SUBTRACT_HEADER
42		FS	FLOATING_NORMALIZE
			65 _FA
			7
			66 _FAN
			7
41		FS	FLOATING_OVERFLOW
			43 FLOATING_ROUND
			10
			65 _FA
			5
			66 _FAN
			5
			68 _FD
			27 32
46		FS	FLOATING_POINT_END
			65 _FA
			12
			66 _FAN
			12
			67 _FM
			26
43		FS	FLOATING_ROUND
			65 _FA
			10
			66 _FAN
			10
			67 _FM
			21
11		FS	GENERATE_SYNCHRONOUS_INTERRUPT
			17 BPP_CHECK
			39 42
			18 IA_SEQUENCE
			30 33
			19 MEMORY_READ
			11
			20 SPP_CHECK
			18 24 30 34

INDEX TO FLOW SEGMENTS

INDEX TO FLOW SEGMENTS

PAGE	LINE	TYPE	NAME AND REFERENCES
			51 _XOR
			2
			52 _ALP
			2 3
			53 _LLP
			4 7
			56 _SLP
			1 2
			57 _SSUM
			1 2
			58 _SDIF
			1 2
			59 _DS
			2 3
			62 DA
			5 6
			63 _DC
			4 5
			65 _FA
			3
			66 _FAN
			3
			67 _FM
			4 5
			68 _FD
			4 5
			8 _LDIF
			2
			79 _ANA
			2
			80 _AA
			2
			81 _LSUM
			2
			88 _SA
			2
			90 _SNA
			2
			91 _SM
			1
			93 _RA
			2
			95 _M
			2
			98 _D
			3 4
			101 _C
			2

INDEX TO FLOW SEGMENTS

INDEX TO FLOW SEGMENTS

PAGE	LINE	TYPE	NAME AND REFERENCES
			69 _XS 4
			70 _IPI 8
			71 _AEI 8
			72 _LIM 6
			73 _IO 9
			77 _LXB 4
			84 _LB 3
			85 _AB 3
			86 _ANB 3
			87 _SB 2
			89 _SXB 4
			100 _CXI 1
			110 _JBNZ 2
			111 _JS 1
			132 [illegible] 2 3
			136 _HCB 2 3
12		FS	GET_IS[illegible]
			5 INTERRUPT_SCAN 8 17
24		FS	GET_SHIFT_AMOUNT
			121 _HLC 2
			122 _HDLC 2
35		FS	GET_SREG
			32 ADD_S 8

INDEX TO FLOW SEGMENTS

INDEX TO FLOW SEGMENTS

INDEX TO FLOW SEGMENTS

PAGE	LINE	TYPE	NAME AND REFERENCES		
			51	_XOR	1
			52	_ALP	1
			53	_LLP	3
			54	_CNT	1
			60	_TSF	4
			61	_DL	2 3
			62	DA	3 4
			63	_DC	6 7
			64	_LBRP	1 2
			67	_FF	2 3
			68	_FD	2 3
			76	_IA	1
			77	_LXB	2
			78	_LDIF	1
			79	_ANA	1
			80	_AA	1
			81	_LSUM	1
			82	_LNA	1
			83	_LM	1
			84	_LB	2
			85	_AB	2
			86	_ANB	2
			92	_BZ	3
			93	_BA	1

INDEX TO FLOW SEGMENTS

INDEX TO FLOW SEGMENTS

PAGE	LINE	TYPE	NAME AND REFERENCES
	40		FLOATING_ADD_SUBTRACT_HEADER
			25
	46		FLOATING_POINT_END
			7 8
	53		_LLP
			12
	54		_CNT
			10
	57		_SSUM
			9
	58		_SDIF
			9
	61		_DL
			4 5
	62		DA
			17 18
	67		_FM
			9 10
	76		_LA
			2
	77		_LXB
			3
	78		_LDIF
			5
	79		_ANA
			5
	80		_AA
			5
	81		_LSUM
			5
	82		_LNA
			3
	83		_LM
			5
	95		_M
			9 10
	98		_D
			6 7
	119		HSC_60
			27
	121		_HLC
			9
	122		_HDLC
			10 11
	123		_HSF
			10 13
	124		_HDSF
			10 11 14

INDEX TO FLOW SEGMENTS

INDEX TO FLOW SEGMENTS

INDEX TO FLOW SEGMENTS

PAGE	LINE	TYPE	NAME AND REFERENCES
			51 _XOR 4
			52 _ALP 6
			53 _LLP 14
80		FS	_AA
85		FS	_AB
71		FS	_AEI
52		FS	_ALP
79		FS	_ANA
86		FS	_ANB
99		FS	_BC
92		FS	_BZ
101		FS	_C
102		FS	_CL
103		FS	_CM
54		FS	_CNT
100		FS	_CXI
98		FS	_D
63		FS	_DC
107		FS	_DJNZ
106		FS	_DJZ
61		FS	_DL
59		FS	_DS
65		FS	_FA

INDEX TO FLOW SEGMENTS

INDEX TO FLOW SEGMENTS

INDEX TO FLOW SEGMENTS

PAGE	LINE	TYPE	NAME AND REFERENCES
50		FS	_MS
48		FS	_OR
93		FS	_RA
27		FS	_REPLACE
			26 REPLACE_CHECK
			5
94		FS	_RI
75		FS	_RP
88		FS	_SA
			89 _SXB
			3
87		FS	_SB
49		FS	_SC
58		FS	_SDIF
56		FS	_SLP
91		FS	_SM
90		FS	_SNA
57		FS	_SSUM
89		FS	_SXB
60		FS	_TSF
51		FS	_XOR
55		FS	_XP
69		FS	_XS

DISTRIBUTION

Defense Advanced Research Projects Agency
1400 Wilson Boulevard
Arlington, VA 23209
ATTN: William Carlson

U.S. Naval Electronics Systems Command
Washington, DC 20360
ATTN: John Machado (Code 330)

Director
U.S. Army TRADOC System Analysis Activity
White Sands Missile Range, NM 88002
ATTN: ATAA-SL (Technical Library)

Commander
Naval Ocean Systems Center
271 Catalina Boulevard
San Diego, CA 92152
ATTN: Russ Eyers (Code 5200)

Defense Technical Information Center
Cameron Station
Alexandria, VA 22314 (12)

Defense Printing Service
Washington Navy Yard
Washington, DC 20374

Library of Congress
Washington, DC 20540
ATTN: Gift and Exchange Division (4)

RADC/ISCA
Griffiss Air Force Base
Rome, NY 13441
ATTN: Armand Vito

DMA/STTP
Bldg. 56, US Naval OBS
Washington, DC 20305
ATTN: Annette Kryqiel

FCDSSA Dam Neck
Virginia Beach, VA 23461
ATTN: Cary D. Upshur (Code 6321) (2)

FCDSSA Code 6
San Diego, CA 92147

NAVSEC
Code 6122
Department of the Navy
Washington, DC 20362
ATTN: Joe Mallonee

NUSC
Code 313
New London, CONN 06320
ATTN: Dr. Charles Arnold

NCR E&M--SD
16550 W. Bernardo Dr.
San Diego, CA 92127
ATTN: Leslie Stevens
Manager, Product Firmware

NANODATA
6065 Madra Ave.
San Diego, CA 92120
ATTN: Robert C. Boe

NANODATA
One Computer Park
Buffalo, NY 14203

Sperry Univac
Univac Park
PO Box 3525
St. Paul, MN 55165
ATTN: Doug Wiedenman
M.S. U2S17

SDC
601 Caroline St.
Fredericksburg, VA 22401 (2)

Hale Associates Research Corp.
PO Box D.J.
Stony Brook, NY 11790
ATTN: John Hale

USC/Information Sciences Institute
4676 Admirality Way
Marina Del Ray, CA 90291

Defense & Space Systems Group of TRW, Inc.
One Space Park
Redondo Beach, CA 90278
ATTN: Barry Press

Naval Sea Systems Command
Washington, DC 20362
ATTN: Lowell Wood

GIDEP Operations Office
Corona, CA 91720

Local:

E41	
K60	
K61 (Carroll Shelton, Henry Walker)	(2)
K70	
K71	
K74	(40)
N30 (R. Hein)	
N52 (Hubbard)	
X210	(6)
X211	(2)